AFRL-IF-RS-TR-2007-165
Final Technical Report
June 2007

A PROBLEM-SOLVING ENVIRONMENT FOR BIOLOGICAL NETWORK INFORMATICS: BIO-SPICE

SRI International

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. M293

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2007-165 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
CLARE D. THIEM Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUN 2007	Final	Sep 01 – Oct 06

4. TITLE AND SUBTITLE

A PROBLEM-SOLVING ENVIRONMENT FOR BIOLOGICAL NETWORK INFORMATICS: BIO-SPICE

5a. CONTRACT NUMBER
F30602-01-C-0153

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)

Patrick Lincoln and Charles John Pedersen

5d. PROJECT NUMBER
BIOC

5e. TASK NUMBER
M2

5f. WORK UNIT NUMBER
93

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SRI International
333 Ravenswood Ave.
Menlo Park CA 94025-3453

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2007-165

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# AFRL-07-0051

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The integration of multiple bioinformatics tools into an integrated suite poses many software engineering, social, and research challenges. We present our work on Bio-SPICE, an integrated open-source framework of tools for modern biology. SRI led the integration tasks, bringing together a community of researchers and practitioners to agree on a set of standards and practices that enabled multiple tools from multiple institutions to be brought to bear on problems of interest to biologists working on DoD-relevant problems. Working through communication and cultural barriers, the Bio-SPICE community developed good working relationships and built a series of impressive use cases, highlighting the effectiveness of combinations of computational tools from multiple sources working together on a common problem. This report documents the efforts at SRI International in leading the integration and combining research efforts across the Bio-SPICE program, including the development of the Bio-SPICE user interface, standards, and community working groups.

15. SUBJECT TERMS
Bioinformatics, Systems Biology, Computational Biology, Software Integration, Open Source Software

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Clare D. Thiem
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UL	142	

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This report presents the results of SRI International's (SRI's) dual efforts as the integrator for and contributor to the Bio-SPICE project under the Defense Advance Research Projects Agency (DARPA) Information Processing Technology Office's (IPTO's) BioComputation program (BioCOMP).

SRI International was the integrator for the Bio-SPICE project. In this role, SRI developed an integrated framework application, the Bio-SPICE Dashboard, and the processes and procedures by which the tools contributed by the other program performers were integrated into this inter-operable suite of systems biology and bio-informatics tools. This report addresses the development and integration process, the Dashboard integration framework, and the development of a web based infrastructure to support tool creation and integration within the community of Bio-SPICE developers. Observations about the program and lessons learned during its execution are included at the end of the report.

SRI International also contributed a number of components to the program, including the Biowarehouse bio-informatics database, the Bio-Mat bridge, a number of smaller integration components, and symbolic analysis tools. Details of these contributions can be found described in this report, along with a listing in Appendix B of all of the contributed components in the program.

A project of this size cannot succeed on the efforts of one or two persons. A large group of dedicated and professional scientists and engineers was essential to making the program as successful as it was. We would like to thank Dr. Peter Karp and his group, in particular Mr. Thomas Lee for making the Biowarehouse a reality; Ms. Valerie Wagner, Mr. Chris Jones, and a number of dedicated SWEP software engineers for spearheading the integration and quality assurance effort; Mr. Derek Artz for being the Bio-SPICE webmaster and web infrastructure support person; Dr. Linda Briesemeister for her championing of the Bio-Mat bridge; and Drs. Carolyn Talcott and Ashish Tiwari for their work extending and modifying a number of symbolic analysis tools to support the Bio-SPICE tool suite. It was indeed a pleasure to work with such a group of consummate professionals.

1 EXECUTIVE SUMMARY/SUMMARY OF KEY ACCOMPLISHMENTS

The Bio-SPICE project, a major portion of the DARPA/IPTO BioComputation Program, began with a vision. Dr. Srikanta Kumar, in collaboration with researchers from top universities and research laboratories around the country, developed a vision of forming a community around a new open-source framework and toolset for twenty-first-century biology. By analogy to the creation of the Simulation Program with Integrated Circuit Emphasis (SPICE) toolset for Very Large-Scale integration (VLSI) engineering, the vision included a core set of common and open interfaces, a robust and easy-to-use infrastructure, and an electronic lab notebook. Over the past few years, the vision has begun to take substantiated form, and through a series of software releases, demonstrations, use cases, reports, research papers, presentations, and community events, Bio-SPICE went from a gleam in the eye to an actual deployed and used system.

A key goal of the Bio-SPICE project was the creation of a framework providing biologists access to the most current computational tools. With the right controls, a biologist is able to use this framework to access data, models, and reports; set up and run computer simulations to test hypotheses or specify and help create protocols for wet-lab experiments; as well as collect, evaluate, and compare data. In the original vision, but not yet implemented, is an electronic notebook to publish and enable comment on repeatable results in an electronic form. Figure 1 illustrates this envisioned process.

To accomplish this, we focused on two key requirements: to develop the framework with a coherent, biologist-friendly user interface and a set of standards that simplify the incorporation of new computational tools. To facilitate the integration of the tools created by other project contributors, we established a continually expanding contribution validation process and a secure community web site to provide on-line support to a growing community of developers and users.

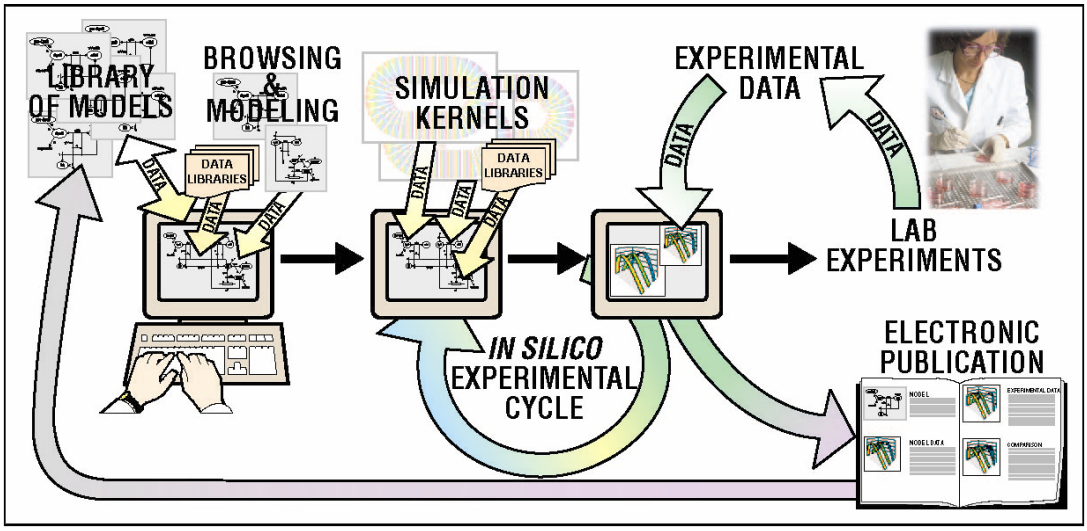

Figure 1. Bio-SPICE Framework.

The project's integrating software platform and graphical user interface is called the Bio-SPICE Dashboard. The Dashboard enables a user to access a growing suite of tools, graphically create workflows that link computational modules together, and access and view data. The Bio-SPICE project produced the Dashboard, a significant and growing number of tools developed by the Bio-SPICE community, and the community web site [1] mentioned above, that provides access to the software as well as a community forum for wide-ranging discussions.

A major accomplishment of this effort was enabling communication, both human and mechanical, in support of the Bio-SPICE vision. Communication between biologists, computer scientists, mathematicians, physicists, chemists, and others is difficult. The communication barrier between biologists and computer scientists is particularly difficult, as they tend to come to a problem with very different viewpoints. Also, communication between computational entities from different computational biology laboratories is difficult. The communication barrier between hand-built custom applications requires massive energy and effort to overcome on a pair-wise basis. A major accomplishment of this effort and other associated researchers over the last five years has been the lowering of these communication barriers. We built a community of researchers including biologists and computer scientists as well as several other disciplines. We also built an agreed set of standard interfaces and languages that enable computational communication between laboratories.

The other major accomplishments were more technical, focused on integration and research. In integration, we created an integration environment to permit the tools developed by other program contributors to be joined together into computational chains of workflows designed to solve specific biological problems. Furthermore, we built and now maintain a bioinformatics database warehouse of key bioinformatics databases with a comprehensive schema that is accessible both to the Bio-SPICE integration environment and as a separate resource. In addition, we created specific tools based upon our research and common needs of the emerging Bio-SPICE community. Using these tools, and together with other Bio-SPICE community members, we performed new creative research to demonstrate uses of Bio-SPICE tools in combination on problems of biological interest and advance the state of knowledge on DoD-relevant issues.

A final task focused on community outreach to build a community of researchers who use the Bio-SPICE tool suite in their research activities.

2 RESULTS

At the end of this effort, functional software can be downloaded from the Web site, a growing suite of tools are accessible through the Bio-SPICE Dashboard, a number of data format standards were created, and interesting workflows were created and executed. The integration tools and infrastructure that were developed facilitate rapid and efficient creation of new computational biology tools.

2.1 THE DASHBOARD

The Bio-SPICE Dashboard provides the user an environment to access software tools. The Dashboard is built upon the NetBeans Integrated Development Environment (IDE), an open source Java software package. The current Dashboard extends the work [2] started in the Arkin

Laboratory, University of California, Berkeley (UC Berkeley) by adding a distributed services discovery and invocation mechanism, as well as several modules to provide additional display components and interfacing components. The Dashboard permits data sources, models, simulation engines, and output displays provided by different investigators and running on different machines to work together across a distributed, heterogeneous network. The source code was freely available to the Bio-SPICE community under the terms of a liberal open source license. It is now available at Sourceforge.net [1].

The ability of the Dashboard to link different components relies on standard exchange formats and application programmer interfaces (APIs), which are described in greater detail below. The Dashboard provides standard display and file manipulation capabilities as well as a standard and a straightforward graphical user interface, so that researchers can concentrate on their research without needing to continually re-invent common components. The Dashboard permits the user to build a simulation workflow, using the available software components, and to save that workflow for later use. The first version of a functional Dashboard was released on May 15, 2003 to Bio-SPICE community members with developer access and was made available to all Bio-SPICE community members in mid-November 2003.

2.2 THE BIO-SPICE COMMUNITY WEB SITE

A distinctive community of researchers formed around the set of Web based infrastructure tools grouped under the label Bio-SPICE Community Web Site. The community began with approximately 100 members, representing primarily the investigators and their colleagues working for DARPA on the Bio-SPICE project. At the end of this effort there were over 1400 registered users.

The community Web site provided members with access to the Bio-SPICE mailing lists, news and announcement capabilities, topical news feeds, on-line documents, hierarchically arranged downloads and resource Web links, discussion forums, file sharing capabilities, wiki pages (html pages editable in a standard browser and used for collaboration) and the ability to browse the project's Concurrent Versions System (CVS) software repositories. Access to site capabilities was based on access levels. Site members had access to some but not all of the site content.

3 INTRODUCTION

3.1 PROJECT GOALS

The central vision of this program was to create a comprehensive networked environment that supported all stages of the research of a molecular/cellular biologist with integrated tools to help the researcher review data, develop models, generate hypotheses, create experiments to explore those hypotheses, run those experiments in a modeling/simulation environment, compare the results of these *in silico* experiments with bench experiments designed to disprove the same hypotheses, and allow the experimenter to iterate over the same experimental domain with great flexibility and control. The intent was that specific tools developed by one researcher would be accessible to all through the integrating framework, allowing researchers to bring a wide range of community developed tools to bear where before he would only have what they had developed themselves. Ultimately we hoped to create a network effect, where the community would build the tools and the tools in term would grow the community.

Our principal technical goal in this program was to create an open-source framework for modeling dynamic cellular network functions and to create a method of organizing biological knowledge around which a user community could develop. We proposed to develop, license, distribute and maintain a comprehensive framework that integrated a suite of analytical, simulation, and visualization tools to aid biological researchers building computable descriptions of cellular functions. Doing this required that we work to create open, transparent standards, open access to the accumulated code base, and promote the integration and re-use of key computational, display and modeling capabilities. For the biologist who just wanted to get on with the research, this approach allows them to use the integrated tool suite without needing to know what goes on 'under the hood.' For the researcher who needs to develop a new tool, the ability to reuse input, output, display, exchange and computational modules from other codes or software projects, there is a significant reduction in software development costs.

Within the scope of these goals, our effort was divided among four primary tasks.

Our first task was to create an integration environment to permit the tools developed by other program contributors to be joined together into computational chains of workflows designed to solve specific biological problems.

Our second task was to build and maintain a bioinformatics database warehouse if key bioinformatics databases with a comprehensive schema that would be accessible both to the Bio-SPICE integration environment and as a separate resource.

Our third task was to create specific tools based upon our research and common needs of the emerging Bio-SPICE community.

Our fourth task was to engage in community outreach to build a community of researchers who use the Bio-SPICE tool suite in their researches.

3.2 APPROACH

3.2.1 Integration Environment

Developing an integration environment and platform that supports Bio-SPICE community designed tools required two subtasks, the development of the platform and the integration of contributor tools into the platform.

3.2.1.1 Development of the Integration Platform

A necessary perquisite to creating the integration platform was for the program contributors to reach agreement on core infrastructure needs, required services, and API's. The initial approach was to attempt a loose coupling of services mediated by the Open Agent Architecture (OAA) framework [3]. The initial Bio-SPICE release used OAA as its sole integration framework. This approach, although entirely satisfactory in some regards, was deficient in others. It was decided that a more tightly coupled solution, employing a UI framework would be more useful. During the December 2002 Principal Investigators (PI) meeting a conceptual mocked up version of a GUI based integration platform was demonstrated. During the subsequent six month development cycle, the first version of the NetBeans based Bio-SPICE Dashboard was developed and released at the end of the cycle as part of Bio-SPICE 3.0. Subsequent development effort went into extending the Dashboard and making it more robust, but the basic SPICE-like visual programming work space where icons representing Bio-SPICE services,

analyzers, data sources, and output displays were 'plugged together' to create complex workflows remained one of the basic paradigms of the Dashboard.

The utility of the Dashboard was tied directly to the quality of the tools that were contributed to work with it, and their ability to intercommunicate. Early in the project, two working groups were established to deal with the issues of exchange formats, the Model Definition Language group (MDL), and the Experimental Working Group (EWG). The MDL focused on exchange formats for simulation and modeling systems, finally settling on Systems Biology Markup Language 2 (SBML2) [4] as an exchange standard. The EWG focused on data exchange formats and initially produced the 'time-series' format. This very simple format proved to be useful in a large number of cases. Later in the program, the EWG endorsed the Minimum Information About a Microarray Experiment (MIAME) format for experimental micro array data.

3.2.1.2 Integrating Contributed Tools and the Software Development Process

While SRI was not under contract to be the primary tool creator, it was tasked to insure that contributed tools would work together within the Bio-SPICE integration environment. To accomplish this, SRI did two things. It established industry standard software development practices for all software that it produced, and made these same processes available to tool developers. These processes included establishing a version controlled repository for source code, an industry standard issue tracker (originally Mantis, later JIRA), an autobuild system for the repository, a support/help desk ticketing system, developers' mailing lists, and web based collaborative work spaces and document repositories. Developer processes included the use of automatically generated API documentation through javadoc and automatic unit testing using JUnit.

To validate contributed tools, SRI established a schedule of delivery tied to the major Bio-SPICE releases, which in turn were tied to the semi-annual PI meetings. Prior to release as a validated tool, each candidate submission would undergo validation testing by SRI software and QA engineers. As time and resources permitted, SRI engineers would work directly with university teams to help them create fully functional, integrated tools.

3.2.2 Bioinformatics Database Warehouse

The BioWarehouse is an open source toolkit for constructing bioinformatics database warehouses using the MySQL and Oracle relational database managers. BioWarehouse integrates its component databases into a common representational framework within a single database management system, thus enabling multi-database queries using the Structured Query Language (SQL) but also facilitating a variety of database integration tasks such as comparative analysis and data mining.

BioWarehouse supports the integration of the following databases: ENZYME [5], Kyoto Encyclopedia of Genes and Genomes (KEGG) [6], BioCyc [7], Universal Protein Resource (UniProt) [8], GenBank [9], NCBI Taxonomy [10], Comprehensive Microbial Resource (CMR) [11], and Gene Ontology [12]. Loader tools, written in the C and JAVA languages, parse and load these databases into a relational database schema. The loaders also apply a degree of semantic normalization to their respective source data, decreasing semantic heterogeneity.

The representational framework is provided in the form of an Extensible Markup Language (XML)-based relational database schema builder, which may be instantiated in either MySQL or

Oracle. In addition to creating the requisite data definition statements, indexes and hyperlinked documentation are also created. The core schema supports the following bioinformatics datatypes: chemical compounds, biochemical reactions, metabolic pathways, proteins, genes, nucleic acid sequences, features on protein and nucleic acid sequences, organisms, organism taxonomies, and controlled vocabularies. It also contains representations for protein interactions and flow cytometry experimental data.

The extended schema incorporates significant extensions for representing MicroArray Gene Expression (MAGE) experimental data. A loader is provided to load data in MAGE format, thus integrating it with other experimental data and/or with databases.

An instantiation of the BioWarehouse has been made available for public use at publichouse.sri.com.

3.2.3 Bio-SPICE Tools

SRI's principal task was to be the program integrator. As we went forward in the program, requirements for utility tools to support the integration framework were identified. The following tools were created to meet those requirements and secondarily to provide examples of well-crafted Bio-SPICE tools for other development teams to use in building their own.

3.2.3.1 BioMat Bridge

Originally called the MATLAB bridge, the BioMat bridge was created to allow Dashboard tools to access the services provided by a local instance of Matrix Laboratory (MATLAB) [13]. This allows researchers to leverage the investment they have made creating models and analytic tools in MATLAB.

3.2.3.2 DBAgent

The DBAgent was written by SRI to provide a well-crafted example of an OAA agent and to provide a simple mechanism for programmatically accessing the BioWarehouse.

3.2.3.3 ptPlot Module

The ptPlot module was created to provide a full featured plotting service for all tool makers. The module was created by providing a custom wrapper for the Open Source ptPlot application developed at UC Berkeley [14].

3.2.4 Bio-SPICE Community, Collaboration and Services

The SRI integration team made extensive use of open source web services and distributed development resources to support the software development efforts of the research teams and to develop a community of researchers.

3.2.4.1 Development Infrastructure

SRI provided a full range of developer support services to the Bio-SPICE community. This included providing a professionally maintained and backed up CVS server, a web browser interface into the repository, an industry standard issue tracking system, and autobuild system, and developer mailing lists specifically tailored to support the Dashboard development and support teams, as well as the EWG and the MDL.

3.2.4.2 Building a Community

The Bio-SPICE web site was one of the principal vehicles used to create and maintain a sense of community among research groups spread across the country. The web site provided a venue to disseminate information, collaborate on research products, and inform the community of items of specific interest. In general, the only access to the web site was through membership. Anyone could request membership, but members who were under contract as BioCOMP performers could access material that was unavailable to general site members.

4 METHODS, ASSUMPTIONS AND PROCEDURES

4.1 PROGRAM PRODUCTS

4.1.1 The Dashboard

The final embodiment of the original project vision of an integration framework is the Bio-SPICE Dashboard. The Dashboard builds upon the visual programming paradigm by providing a work space in which iconic representations of data sources, analyzers, simulations engines and display modules can be visually chained together into workflows. The following describes how the Dashboard is used.

4.1.1.1 Running the Dashboard

Windows users should find shortcut links on their desktop to the Dashboard, the OAA facilitator, and the Dashboard Uninstaller. To run the Dashboard, locate the Dashboard's bin directory.

- On Windows, this is most likely: `C:\Program Files\Bio-SPICE\Dashboard[Version #]\Dashboard\bin`
- On Linux, this is most likely: `/opt/Bio-SPICE/Dashboard2.0.0/Dashboard/bin`

The executable to run the Dashboard is named "runide"

- Windows users: double-click on `runidew.exe`. There is also `runide.exe`, which starts a console window in addition to the Dashboard. Debugging and/or information messages are sometimes printed to this console window.
- Linux users: run `./runide.sh`
- Mac users: double click the Bio-SPICE icon. A small window, the Bio-SPICE Launcher, will appear. By clicking the "Launch Bio-SPICE" button, the dashboard will be launched.

4.1.1.2 Introduction to Workflows

A workflow is an acyclic graph representing a high-level task that a user wishes to run. The individual parts of this task are all the nodes in a workflow, and consist of all the modules that will be run and the data they will be analyzing or producing.

A workflow may contain source documents, destination documents, and analyzers. A source document represents data read from a file. Similarly, a destination document represents data being written to a file. Analyzers may have any number of inputs and/or outputs.

For a workflow to be valid, all required inputs and outputs from all nodes (documents and analyzers) must be satisfied. In addition, all source and destination documents must have a file

associated with them. Analyzer inputs and outputs may be satisfied by connecting links to other documents and analyzers of matching type. In addition, some analyzer inputs (for example, text) may be satisfied by manually editing the input parameters. (See section 4.1.1.4 More About Workflows for instructions on how to manually edit input parameters.)

4.1.1.3 Building a Simple Workflow

Users were encouraged to make sure they had downloaded the latest Bio-SPICE modules using the Update Center that was operational during the life of this effort. This process is described in the installation instructions in the Dashboard User Manual which can be found by clicking the Help Contents button under the Dashboard's Help pull down menu.

- **Open the Workflow Editor** window by choosing "Open Workflow Editor" from the "Bio-SPICE" menu or using the shortcut CTRL-T. Figure 2 provides an example of what appears when the editor is opened. Note: The Dashboard can operate in single-document or multiple-document mode. If the Workflow Editor opens as a separate window from the Dashboard, you can "attach" the window back to the Dashboard by choosing "Window" --> "Attach MDI frame" --> "Right".

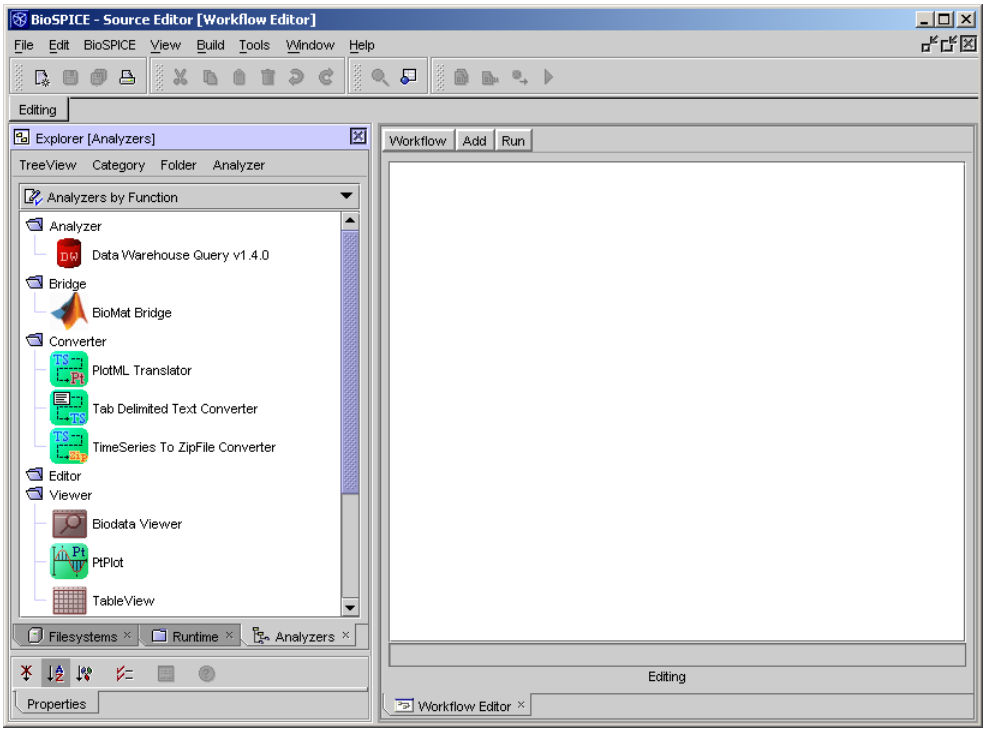

Figure 2. The Dashboard Workflow Editor.

- **Add a source document.** Click on the "Add" menu button and select "Document" --> "Source". This places an empty source document icon on the Workflow Editor. To associate the source document with a particular file, right-click on the "Doc" icon and select "Edit" (as shown in Figure 3) or double-click on the icon. From the file chooser that appears, load a sample document. In this case, load a sample file that has the `.timeseries` extension. (Alternatively, you can drag a `timeseries` file from a directory mounted in the Dashboard

Filesystems Explorer window and drop it into the Workflow editing area.) An example TimeSeries data file is available in the *Using the Dashboard* section of the Dashboard User Manual.

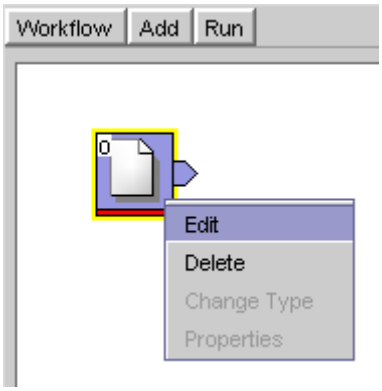

Figure 3. Add a Source Document.

- **Add an Analyzer.** Click once on the "TableView" analyzer to select it. Then drag-and-drop the analyzer onto the workflow area or use the right-click context menu item "Add to Workflow". Both the source document and the analyzer will have red bars under them, as illustrated in Figure 4, indicating that their inputs and outputs have not all been satisfied.

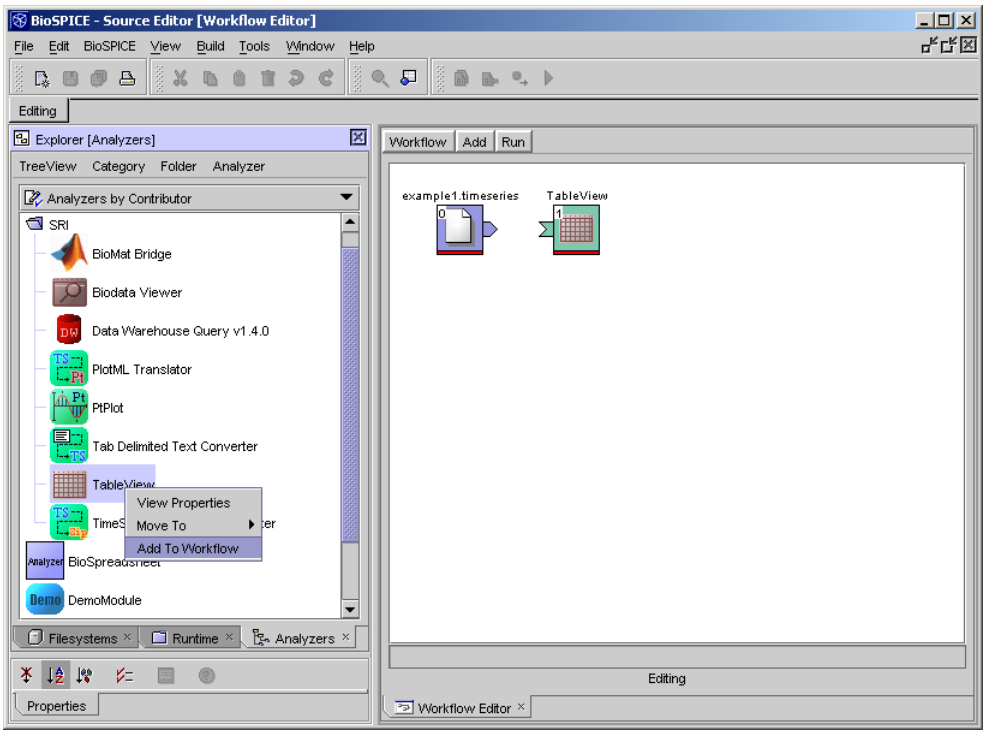

Figure 4. Add an Analyzer.

- Connect the output of the source document to the input of the analyzer by clicking on the output polygon on the source document and dragging a connection line to the input polygon of the TableView Analyzer. Because the source document has one output (a TimeSeries object) and the Analyzer has one input (also a TimeSeries object), the workflow is valid and all status bars and connection links turn green as shown in Figure 5.

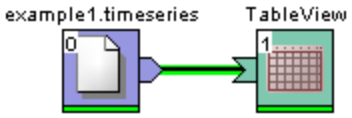

Figure 5. Connecting two Workflow Objects.

- **Run the workflow.** Click on the "Workflow" menu button and select "Start". When running the workflow, the background color changes to light gray to symbolize that editing is not possible at the moment. In addition, the status message at the bottom turns from "Editing" to "Running..." Each node in the workflow is highlighted as it is executed. When the workflow execution halts, the TimeSeries data stored in the source document will be displayed in a table like that shown in Figure 6.

TimeSeries Tables [Table 0]

time	temperature	conc1	conc2
0	20	-45	-125
0.1	20	-44	-122
0.2	20	-43	-119
0.3	20	-42	-116
0.4	20	-41	-113
0.5	20	-40	-110
0.6	20	-39	-107
0.7	20	-38	-104
0.8	20	-37	-101
0.9	20	-36	-98
1	20	-35	-95
1.1	20	-34	-92
1.2	20	-33	-89
1.3	20	-32	-86
1.4	20	-31	-83

Table 0

Figure 6. Workflow Output.

4.1.1.4 More About Workflows

4.1.1.4.1 Saving, Loading, and Clearing

Once a workflow is created it can be saved for later use. After creating your workflow, choose "Save" from the "Workflow" menu. In the file save dialog, enter the name of your workflow followed by an `.xml` extension. A description of your current workflow will be saved to a file in XML format.

To load a previously saved workflow, choose "Load" from the "Workflow" menu. In the file chooser dialog, select a workflow file. This will load the workflow into the Workflow Editor.

To delete all the nodes (documents and analyzers) currently on display in the Workflow Editor, choose "Clear" from the "Workflow" menu.

4.1.1.4.2 Reviewing and Manually Editing Input Parameters

A user can review and manually edit an input parameter by right-clicking on an analyzer and choosing "Edit". (Alternatively, double-click on the analyzer.) An input parameter editor window will appear. Figure 7 shows an example of the input parameters of the "Table View" analyzer in the sample workflow.

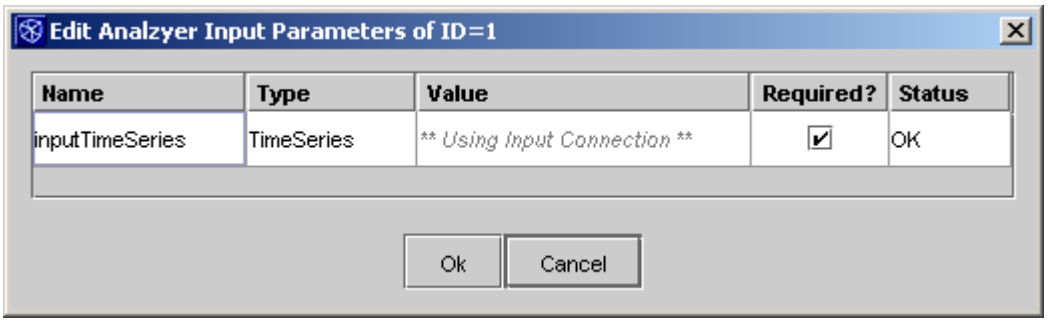

Figure 7. Workflow Output.

If you delete the workflow source document or the connector (using right-click menu on either), the input parameter dialog will look like Figure 8. Because the required input is of type Biodata, it cannot be manually edited.

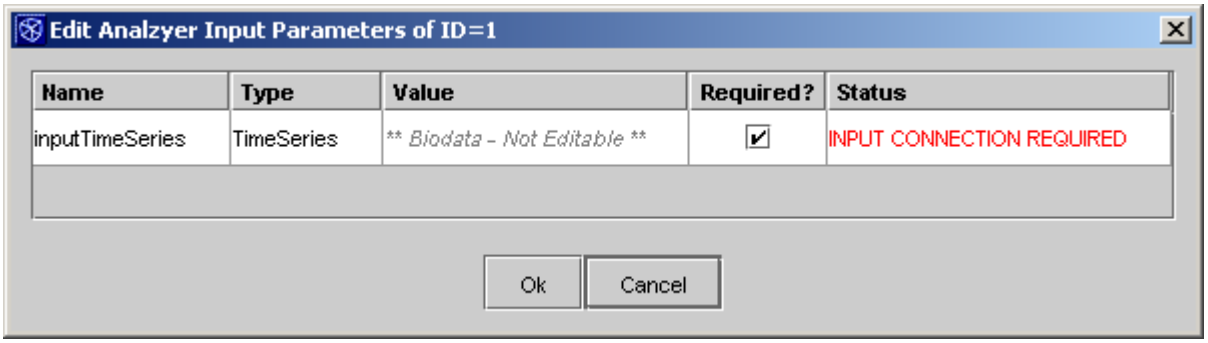

Figure 8. Display Input Parameter Error Status.

To see an example with input parameters that are actually manually editable, drop a "Data Warehouse Query" analyzer into the workflow. When you edit the analyzer (right-click or double-click), it will show the editor for input parameters similar to Figure 9 below. Edit the required or not-required values. Note that supplied values are surrounded by double-quotes to symbolize that they are stored as Strings. When the dialog is closed, the system tries to convert the supplied values to the given types and will alert the user if it cannot do so.

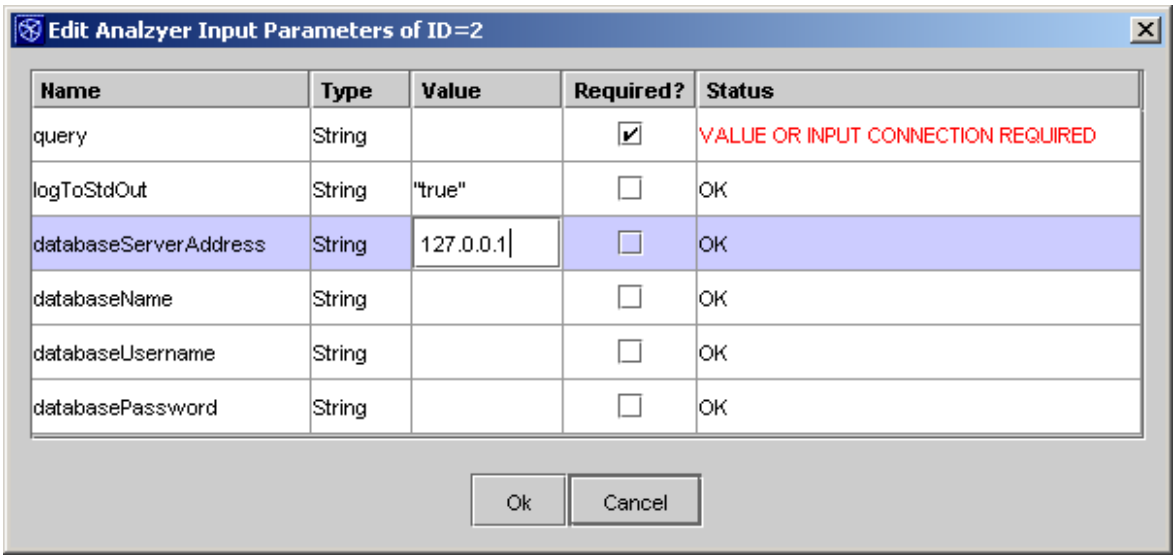

Figure 9. View Analyze Input Parameters.

4.1.1.4.3 Reviewing and Manually Editing Connector Parameter Mappings

To manually edit the mapping of output to input parameters of a connector, right-click on the connector and choose "Edit". (Alternatively, double-click on the connector.) A connector parameter mapping editor window will appear. This editor window will also pop up if the connector has more than one possibility of mapping outputs to inputs. Then, the user needs to add the mapping of their choice by highlighting the respective rows in the tables with unused parameters and clicking the button "Add Mapping". Figure 10 shows the mapping of the example workflow above.

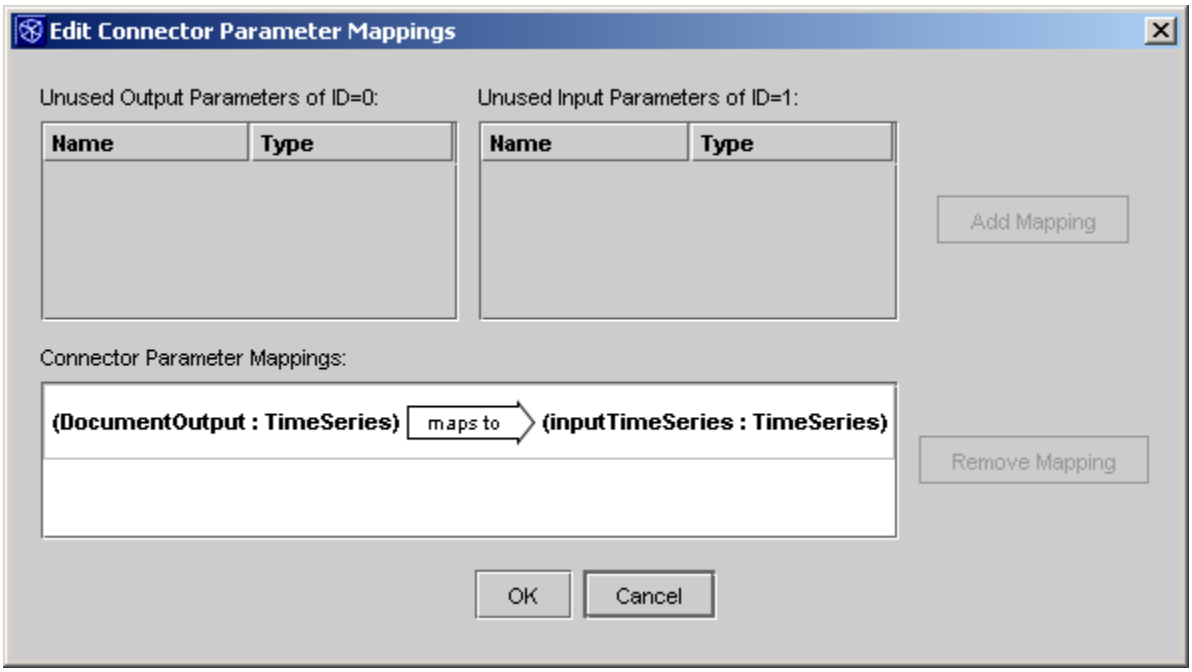

Figure 10. Editing Connector Parameter Mappings.

4.1.1.4.4 Viewing Analyzer Properties

To view more information about an analyzer chose the menu item "Properties" from the right-click context menu of an analyzer placed in a workflow. Alternatively, one can also view the properties from the analyzer tree view as described in the *Using the Analyzer Tree View* section of the Dashboard User Manual. The window with the analyzer properties floats on top of all other windows but allows the user to continue editing the workflow. See Figure 11 below for an example of the properties for the "Data Warehouse Query" analyzer.

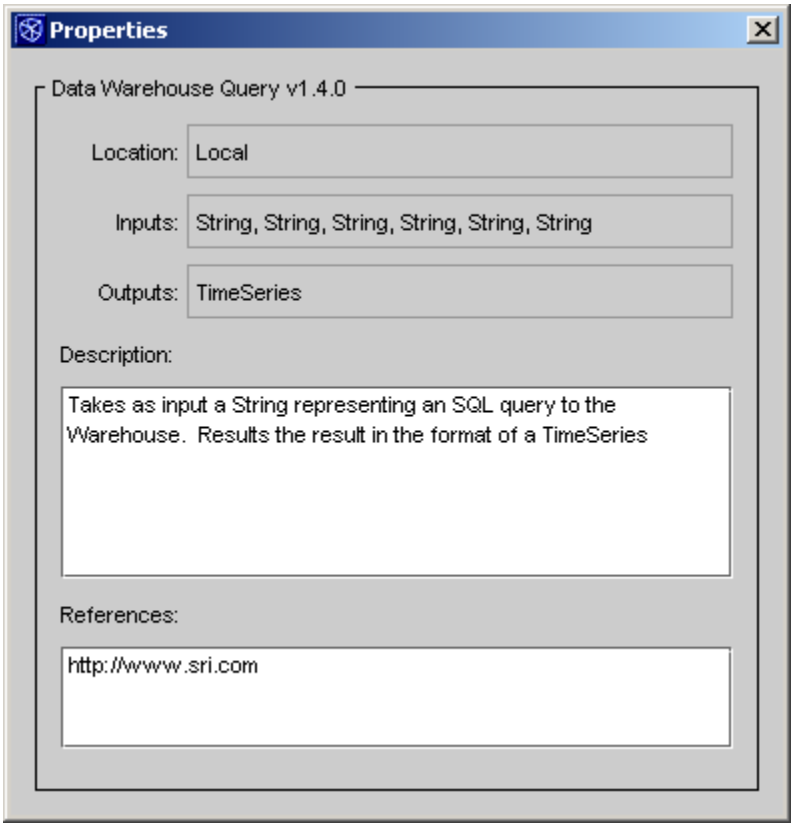

Figure 11. Analyzer Properties Pane.

4.1.1.4.5 Viewing and Changing Document Properties

A user can view the properties of a workflow document that has been assigned to a file by right-clicking on the document and select "Properties". For the document used in the example, the same TimeSeries data file previously referred to in section 4.1.1.3, the properties look like the screenshot in Figure 12.

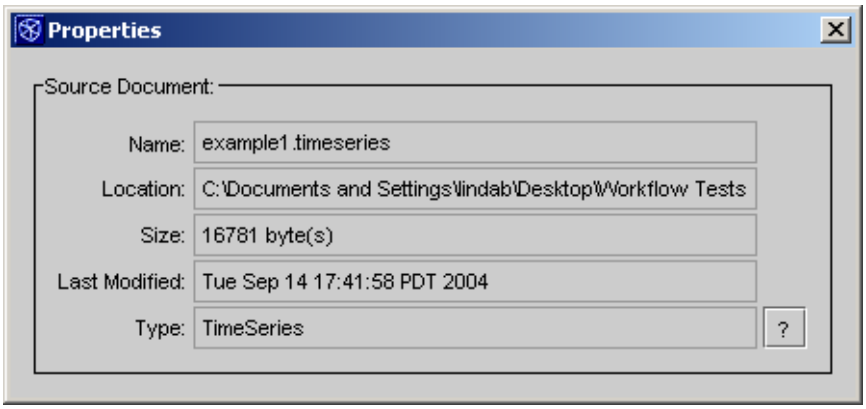

Figure 12. Workflow Document Properties.

For more information about the type of the document within the dashboard, click the "?" button at the bottom. A new window similar to Figure 13 appears.

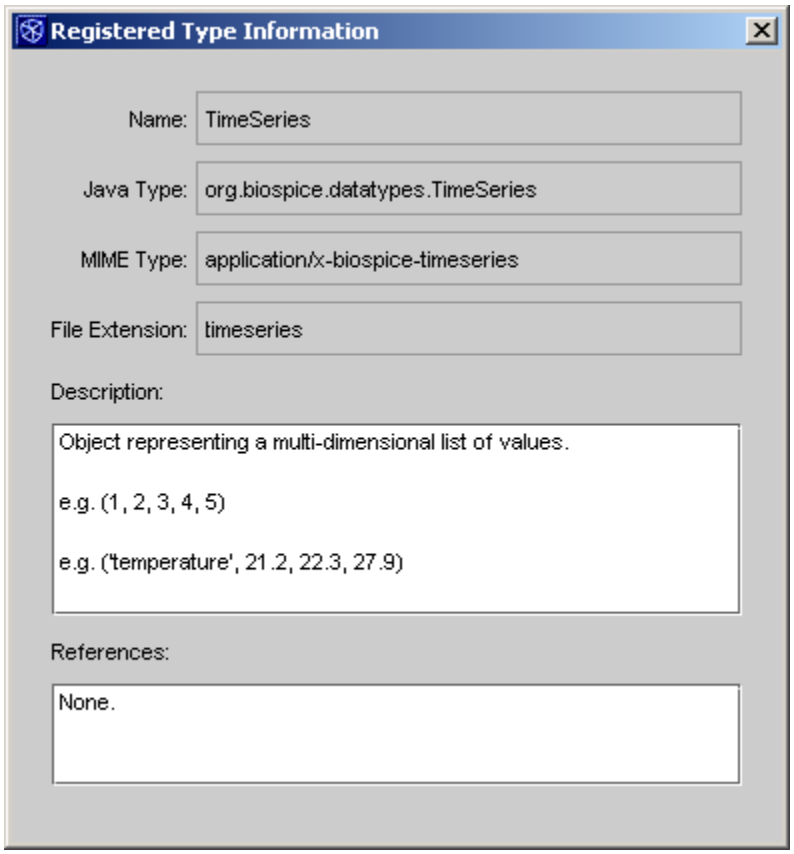

Figure 13. Workflow Document Type Properties.

As of version 1.4 of the Bio-SPICE Dashboard Core Module, users can override the type of a source document. As an example, build a small workflow using the TimeSeries data file from section 4.1.1.3 for a source document, and attaching this source document to a "Biodata Viewer" analyzer as seen in Figure 14 below. Chose the menu item "Change Type" from the source document's context menu.

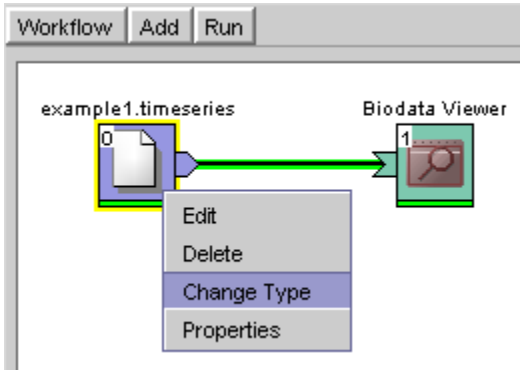

Figure 14. Changing Document Type.

In the dialog that appears, choose for example "Text File" from the combo box at the bottom, as shown in Figure 15, and click "OK". When re-opening the dialog to change the type, it shows that the current type is "Text File" rather than the resolved type, which is "TimeSeries" for this document.

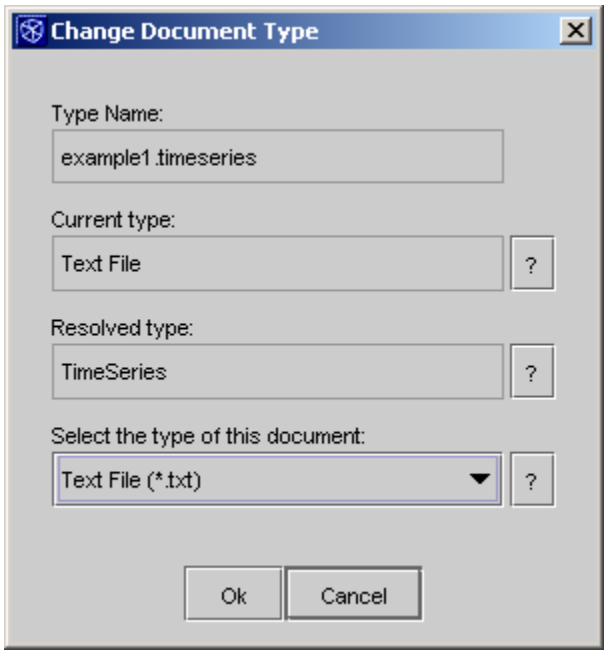

Figure 15. Document Type Edit Pane.

After running this workflow, the "Biodata Viewer" shows the contents of the file in a new window similar to Figure 16 below (note that you can wrap the lines using the check box). Try to attach the source document with the altered type to a "TableView" analyzer; it is not possible as the "TableView" analyzer requires a TimeSeries input but we had just overridden the type to be Text File.

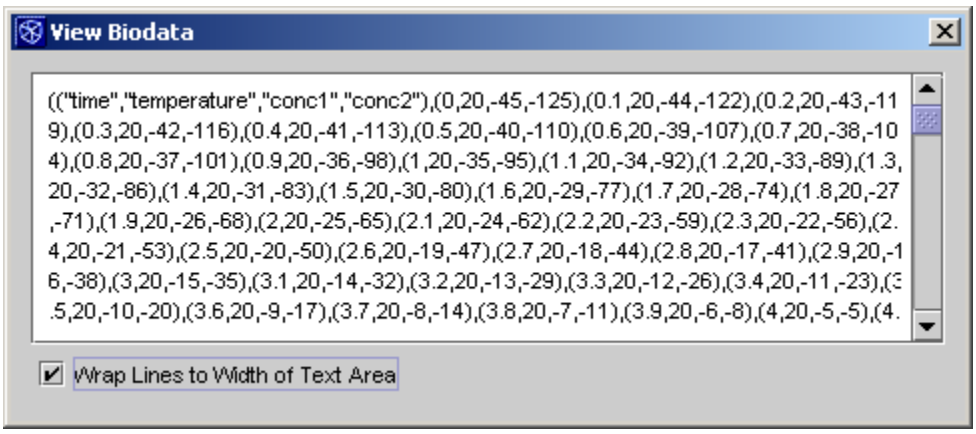

Figure 16. Biodata Viewer Screen Shot.

4.1.1.5 The Dashboard Architecture and API

4.1.1.5.1 Introduction

With the use of common datatypes (SBML, TimeSeries, and EWG) and communication frameworks (OAA, NetBeans, and Systems Biology Workbench (SBW)[15]), it becomes possible for Bio-SPICE tools to truly interoperate.

This interoperation occurs through the Dashboard, the program with which Bio-SPICE tools communicate. The Dashboard sends data and computation requests to tools and can save those tools' results to files or route them to yet other tools. In order to integrate tools, the Dashboard needs a minimum of information about them: what kinds of data they accept, what functions they perform, and what kinds of data they return. Bio-SPICE tools provide all this introspection information via the Bio-SPICE API.

This document specifies this API. There are two main parts to this document: the first describes how Bio-SPICE tools specify the datatypes they use as inputs or outputs. The second part addresses the different ways in which tools can communicate and integrate with the Dashboard.

4.1.1.5.2 Datatypes

The Bio-SPICE API uses the Multipurpose Internet Mail Extensions (MIME)-type datatype naming scheme. MIME is a flexible, extensible naming scheme, widely in use on the Internet and in some operating systems for providing clues to browsers and mail readers as to how to interpret different types of data.

Essentially, there is one MIME name per datatype. The name consists of a type and a subtype, separated by a forward slash (/). For example, XML text has the MIME-type text/xml and JPEG image files have the type image/jpeg. Detailed MIME specifications are available on the internet [16].

The Bio-SPICE API specifies the (MIME) names and formats of a few, widely used biological data formats, as well as widely used primitive types (like ints and strings). Many MIME types in use on the Internet have also been defined. Tools may also define their own types if they deal with data that does not have a currently defined datatype. How to do this is described in section 2.3.

4.1.1.5.2.1 Biological Datatypes

Bio-SPICE currently supports SBML level 2, a TimeSeries format, and various experimental data types (the data are experimental, not the types). Other datatypes will likely be added in the future; these may start as custom datatypes before becoming core types. The currently supported datatypes were "blessed" by the System Engineering and Product Design Task Force (SEPDTF), EWG, or MDL committees, and supported internally by the Dashboard. All of these types have unique MIME-types, defined in the Table 1 below.

Table 1. Biological Data MIME-Types.

Description	MIME-type
SBML Level 2	application/sbml+xml
Time Series	application/x-Bio-SPICE-timeseries; charset=utf-8
General EWG	application/x-ewg+xml
Cellular Spatial Information	application/x-cellular-spatial+xml
Protein Expression	application/x-protein-expression+xml
Behavioral	application/x-behavioral+xml
Chemical Structure	application/x-chemical-structure+xml
Gene Expression	application/x-gene-expression+xml
Biomolecular Sequence	application/x-biomolecular-sequence+xml
Metabolite Expression	application/x-metabolite-expression+xml
Molecular Activity	application/x-molecular-activity+xml
Molecular Interaction	application/x-molecular-interaction+xml
Genetic	application/x-genetic+xml
Population Heterogeneity	application/x-population-heterogeneity+xml
Phenotype	application/x-phenotype+xml
Histopathology	application/x-histopathology+xml
Experimental Protocols	application/x-experimental-protocols+xml
Literature	application/x-literature+xml

(Note, the +xml suffix on the MIME-type indicates that the data is in XML format. Generic XML data, however, uses the MIME-type application/xml. Also, the "x-" prefix on the subtype indicates that the particular format is somewhat experimental and has not been registered with the IANA, the official MIME-type registry organization.)

4.1.1.5.2.2 Primitive Datatypes

In addition to complex biological data, many tools accept primitive data such as strings, ints, and floating-point numbers. These types are listed in Table 2.

Table 2. Primitive Data Types.

Description	MIME-type
Boolean	application/x-Bio-SPICE-boolean
Double	application/x-Bio-SPICE-double
Int	application/x-Bio-SPICE-int
String	application/x-Bio-SPICE-string

4.1.1.5.2.3 Custom Datatypes

Tools that manipulate novel datatypes may define their own types. Registering a new datatype is as simple as creating a new MIME-type name.

For instance, consider a theoretical tool that performs bacterial quorum sensing simulation given the coordinates and orientation of bacteria within a colony; it could indicate this coordinate and orientation data as an input to the Dashboard by marking its input as being of type "application/x-Bio-SPICE-quorum-coordinates." Alternatively, it may be called "text/x-quorum-coordinates+xml" if it is an XML file.

Generally, the developer of a custom type has great latitude in the naming of the MIME-type. The only firm requirement is that the MIME-type name be unique. There are a few important conventions to follow, though, including prefixing the subtype of new/experimental formats with "x-" and using existing MIME-types when appropriate. Complete lists of conventions for MIME-type naming are defined in Request for Comment (RFC) memorandums 2046 [17] and 2048 [18].

It is generally a bad idea to invent your own data type. As a general rule, new formats should be developed ONLY if the data a tool needs/produces cannot be represented in any of the existing types. In any case, it is much better to work with the BioCOMP MDL, EWG, and SEPDTF committees to change existing data formats to accommodate additional data than to formulate new formats.

Those implementing native Dashboard tools may provide additional support for custom datatypes. This includes providing parsers, templates for sample/empty files, and allowing the Dashboard UIs to recognize files containing custom datatypes. More on this can be found in the forthcoming Bio-SPICE Developer Guide.

4.1.1.5.3 Dashboard Integration

Several methods of integration into the Dashboard are possible.

OAA-style integration is recommended for tools that already export functionality via OAA. This method is by far the easiest way to integrate existing OAA agents into the Dashboard, and is discussed in section 4.1.1.5.3.1.

Direct Dashboard integration is recommended for tools written in Java that wish to integrate tightly with the Dashboard. There are both efficiency and functionality advantages to native Dashboard integration. These are discussed in further detail in section 4.1.1.5.3.2.

Tools that are not already OAA agents or which may not be suitable for native integration (e.g. if the tool is non-java or if a quicker integration method is desired) may integrate via an XML descriptor file, which is described in section 4.1.1.5.3.2.2.

All of these methods are essentially different ways of informing the Dashboard about the functions that a tool performs and the data that each function can take in and return. The conceptual information provided by each method is equivalent; only the syntax differs. (The native Dashboard API is an exception; it also provides hooks for more sophisticated functionality and thus is a superset of the information conveyed by the other methods.)

Conceptually, the information provided through the API is the following for each tool function:

- The function's name
- A list of parameters the function accepts as input
- A list of arguments the function returns as output
- A function description
- References and help for function use

For each parameter/argument, the following information is provided:

- Its name
- Its MIME-type (as per section 4.1.1.5.2)
- Whether it is an input parameter or an output argument
- Whether the parameter is an array of values of the given type. For each input parameter the following additional information is provided:
- Whether it needs to be assigned a value in order for the function to operate successfully.
- Optionally, it may specify a set of acceptable values.
- Optionally, it may specify a default value. For each output argument the following additional information is provided:
- Whether it will be present at the completion of every successful function run.

Functions may define input parameters, output arguments, or both. They may also accept no input parameters or return no output arguments, though functions with neither input nor output are not allowed.

Below we describe the syntax for specifying function and parameter information for each integration method.

4.1.1.5.3.1 OAA API Syntax

The OAA integration method is the easiest way to integrate existing OAA agents into Bio-SPICE. It requires the augmentation of the agent declaration with additional syntax. This syntax expands on the vanilla OAA syntax used in previous Bio-SPICE versions to include information on both the mutability and type of each argument to each solvable of an agent. The mutability can be used to distinguish an parameter from a return argument, and it is expected to be one of IN or OUT. The type is one of the types as per section 4.1.1.5.2 of this document. In addition, this OAA syntax includes a brief description and list of references for any given program; these fields are assumed to be of great importance for scientific software. The syntax definition follows; non-literal strings are given in angle brackets.

```
[solvable(<solvable_name>(<arg_list>),[argtypes(<arg_type_list>),
description(<description>),references(<references>)])]
```

```
   <solvable_name> is the name of this functor.

   <arg_list> is a comma-separated list of argument variables.  These
       would typically be Titlecase words.

   <arg_type_list> is a comma-separated list of argument types,
       corresponding to <arg_list> above.  Each entry takes the form
       <inout>(<type_name>,<reqDet>), where <inout> is either in or
       out, and <type_name> is one of the types defined above.  The
       <reqDet> parameter is a boolean, and its meaning depends on
       the value of <inout>.  For in parameters, <reqDet> represents
       whether the given parameter is required; a non-required
       parameter is assumed to have a default value.  For out
       parameters, <reqDet> should generally be true; see the Typed
       OAA Specification for details.

   <description> and <references> are both HTML strings providing
       more information about the functor.  Remember to use HTML
       specials like &amp; and &lt; for characters like & and <.
```

An example solvable declaration (broken into multiple lines for readability) is:

```
[solvable(simbug(Model,Step,Length,Results),

   [argtypes(in('application/x-sbml2+xml',true),
       in('application/x-Bio-SPICE-double',false),
       in('application/x-Bio-SPICE-double',true),
       out('application/x-Bio-SPICE-timeseries',true)),

   description('SimBug simulates a bacterium.'),

   references('<a href="SimBug"

target="l">http://www.simbug.com/">SimBug home page</a>')])]
```

4.1.1.5.3.2 Native Dashboard Integration API

The Bio-SPICE Dashboard is built on top of the NetBeans platform, an open-source, Java-based framework that can be used to integrate varied modules and provide prepackaged built-ins for common application functionality (such as network updates for new versions, filesystem explorers, and the like). The Dashboard itself is a configured, branded version of the platform with additional NetBeans modules installed. Native Dashboard Bio-SPICE tools are essentially NetBeans modules that conform to a few additional requirements.

The native Dashboard integration method is only recommended for Java applications (non-Java applications may use this method, but it will require the wrapping of the application in Java code). A native Dashboard integration is more complicated than augmenting the OAA declaration syntax for an existing OAA agent, but it does allow for many functional and efficiency advantages.

Functional advantages to a native integration include:

- Dashboard recognition, editing, and templates for files with custom datatypes

- The option to function within either MDI or separate window modes
- Custom addition of toolbar buttons and menu items to the Dashboard
- Integrated installation with the Dashboard
- Network-enabled version updates through the Dashboard
- More integrated help and documentation functionality
- Persistent storage of user settings
- Built-ins for internationalization/resource bundles

Native Dashboard tools (and the Dashboard) all run within a single address space and enjoy all the resulting efficiencies:

- Reduced context switch time
- Smaller memory footprint
- Object sharing between native tools
  - Further reduces memory footprint
  - Cuts down on reparsings of data documents
  - Avoids sending data across many software and hardware layers when sending objects between native tools (as would be required to send data over a network)
- Native Dashboard tools may be enabled or disabled at runtime, further reducing memory footprint.

Tool functions and their parameters are specified by extending the class `org.Bio-SPICE.analyzer.AnalyzerFactory` and implementing the interface `org.Bio-SPICE.analyzer.Analyzer`. These are critical classes that provide introspection "hooks" to the Dashboard. Classes extending `org.Bio-SPICE.analyzer.AnalyzerFactory` and implementing `org.Bio-SPICE.analyzer.Analyzer` are registered in the Dashboard via a layer file, described in the following section. How to implement the additional integration functionality listed above is discussed in the Bio-SPICE Developer Guide which can be found by clicking the Help Contents button under the Dashboard's Help pull down menu.

Examples of Bio-SPICE NetBeans modules can be found in the Dashboard source tree, in the sbmlsupport and oaa-bridge directories, among others.

The rest of section 4.1.1.5.3.2 covers the files and file structures necessary to create in order to register AnalyzerFactory and Analyzer classes with the Dashboard.

4.1.1.5.3.2.1 Essential bookkeeping and administrative integration files

A native Dashboard tool is contained within a NetBeans NBM file. This file is a zip archive that must conform to a relatively simple directory structure specification [19].

Within the NBM file, there will be at least one Java Archive (JAR) containing the Java code for the Bio-SPICE tool. That JAR needs to contain a manifest file with bookkeeping directives for the Dashboard. These directives include the module name, an optional description, dependencies on shared JARs and NetBeans/Dashboard modules, and a pointer to a layer file. Native Dashboard tools must include a dependency on the Dashboard "module." It is specified with the following line:

```
OpenIDE-Module-Module-Dependencies: org.Bio-SPICE.ide/1
```
More information on the manifest format and the module JAR file can be found online [20].

The layer file referenced in the manifest is an XML file that, among other things, registers tools' AnalyzerFactory and Analyzer classes in the NetBeans lookup system. The Dashboard accesses the lookup system to obtain references to native Dashboard tools' functions. More information on the lookup system can be found online [21].

An example of a layer file for a Bio-SPICE tool would look like:

```
<filesystem>
  <folder name="Services">
    <folder name="Hidden">
      <file name="edu-myuniversity-Bio-SPICE-
MyBigAnalyzerFactory.instance">
        <attr name="instanceOf" stringvalue="org.Bio-
SPICE.analyzer.AnalyzerFactory"/>
      </file>
    </folder>
   </folder>
</filesystem>
```

This file indicates to the Dashboard that the tool is registering exactly one function, which is implemented in the class edu.myuniversity.Bio-SPICE.MyBigAnalyzerFactory.

Please note the use of dashes instead of periods to separate package names in the class name. Additional directives that can be added to the layer file to support extended functionality are described in the Bio-SPICE Developer Guide which can be found by clicking the Help Contents button under the Dashboard's Help pull down menu.

4.1.1.5.3.2.2 General tool integration (XML-wrapped)

This last method is the most general integration method for integrating tools into Bio-SPICE. It will allow for the integration of tools written in any language, and any implementation style, whether it is command-line, web-based, or based on TCP/IP socket communications. Although it could also be used to wrap OAA agents, existing OAA agents are better wrapped using the OAA-specific method described in section 4.1.1.5.3.1 is than the general method described in this section.

Important implementation/support note: Support for TCP/IP based tools and for passing more than one biological datatype via standard in (command line tools) or an HTTP post is dependent on the specification of a transport format needed to include multiple documents/parameters in a single transmission. This specification is pending completion; as such, please only use this method for HTTP or command-line tools, and those only accepting a single non-array biological parameter (and any number of primitive parameters) or returning a single, non-array biological or primitive argument.

All the core function and argument information described in section 4.1.1.5.3 is specified in a relatively simple hierarchy of XML elements nested in an element named "service". The additional information - about location and access method of the tool - is included in an element named "location." Here is an example of an HTTP-based simulator:

```
<?xml version="1.0" encoding="UTF-8" ?>

<n:analyzer analyzerName="edu.myuniversity.mysimulator"

analyzerReadableName="A generic simulator" xmlns:n="http://www.biospice.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.biospice.org analyzer.xsd">

  <service serviceName="simulate">

    <input parameterName="startTime" type="application/x-biospice-double"
required="true" />

    <input parameterName="endTime" type="application/x-biospice-double"
required="true" />

    <input parameterName="numPoints" type="application/x-biospice-int"
required="true" />

    <input parameterName="selectionList" type="application/x-biospice-string"
array="true" required="false" />

    <input parameterName="model" type="application/x-sbml+xml"
required="true" />

    <output parameterName="result" type="application/x-biospice-timeseries"
required="true" />

  </service>

  <location>

    <HTTP URL="http://aserver.somewhere.edu/simulator" />

  </location>

</n:analyzer>
```

This sample tool has one function, "simulate," which accepts an SBML model and various configuration parameters (such as the start and end times of the simulation), and returns a single timeseries datum. The URL at which it can be accessed is specified in the location section.

The complete specification of this XML format appears below.

4.1.1.5.4 API Worksheet and Primitive Datatype Formats

The following were developed to assist tool developers.

4.1.1.5.4.1 API Worksheet

This worksheet is to aid in crafting your tool's Bio-SPICE integration API and can be used to submit your API for validation.

Developer and Group:

1. Tool name:

2. Short tool description:
3. How is your tool implemented? Please indicate both programming language used (C, Java, Lisp, etc) as well as implementation style (web-based, command-line, client-server, etc.).
4. Function 1:
    a. What is the function's name? (May be omitted if tool has only one function)
    b. What provide a short description for the function. (May be omitted if tool has only one function)
    c. Please describe the inputs to your tool. If there are none (tool is a "source"), please write "none" here:
        i. Input 1
            1. Name:
            2. Type (Please use one of the MIME-types listed in the tables of section 2 of this document. If you require a novel type, please indicate its MIME-type and explain why an already defined type cannot be used):
            3. Is it an array of this type?
            4. Does the tool require this input for the tool to function?
            5. Is there a limited set of legal values for this input? If so, please list here.
            6. Is there a default value for this input? If so, please specify:
        ii. Inputs 2+. Repeat section 5.c.i for all remaining inputs
    d. Please describe the outputs of your tool. If there are none (tool is a "sink"), please write "none" here:
        i. Output 1
            1. Name:
            2. Type (Please use one of the MIME-types listed in the tables of section 2 of this document. If you require a novel type, please indicate its MIME-type and explain why an already defined type cannot be used):
            3. Is it an array of this type?
            4. Does the tool always return this output at the end of a successful execution?
        ii. Outputs 2+. Repeat 5.d.i for all remaining outputs
    e. Please list published or online references for this function, including help and other documentation:
5. If your tool has additional functions, repeat section 5 for each function

4.1.1.5.4.2 Primitive Type Formats

This section lists definitions for the legal representations of Bio-SPICE primitive types. Non-primitive biological types are defined in a number of online documents such as those found at the SBML website [4].

Formats for primitive types are taken from the XML schema definitions (XSD) for primitive type formats. (Note, only a small subset of these XML schema primitive types are legal for Bio-SPICE.) The definitions of all XSD primitive types can be found on the internet [22].

The names of the XSD type to which each Bio-SPICE type corresponds is listed below in Table 3. A direct link to the type's format specification is also listed.

Table 3. Bio-SPICE Primitive Type Formats.

Type	Boolean
MIME type	application/x-Bio-SPICE-boolean
Format	http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#boolean
Type	**Double**
MIME type	application/x-Bio-SPICE-double
Precision	IEEE 754 "double format" precision
Format	http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#double
Type	**Int**
MIME type	application/x-Bio-SPICE-int
Range	-2^{31} to $2^{31}-1$
Format	http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#int
Type	**String**
MIME type	application/x-Bio-SPICE-string
Format	http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#string

4.1.2 The BioWarehouse

[The following description of the Biowarehouse taken from a paper published by Dr. Peter Karp which can be found at this URL: http://www.biomedcentral.com/1471-2105/7/170. For footnotes and other references in this section (preceded by Lee et al.), please refer to the original source.[23]]

The bioinformatics database warehouse task addressed the problem of interoperation of heterogeneous bioinformatics databases.

4.1.2.1 Background

The importance of the database (DB) integration problem to bioinformatics has been recognized for many years [Lee et al. 1-6]. Many questions in bioinformatics can be addressed only by combining data from multiple DBs, and DB integration also permits crosschecking of individual DBs for validation. In 1993, a summary of a U.S. Department of Energy workshop on genome informatics noted that "achieving coordination and interoperability among genome DBs and other informatics systems must be of the highest priority. ... We must begin to think of the computational infrastructure of genome research as a federated information infrastructure of interlocking pieces..." [Lee et al. 7].

One approach has involved mediator-based solutions that transmit multidatabase queries to multiple source DBs across the Internet. Although some progress has been made in developing mediator technology, we argue that these systems face several practical limitations (see Section "Comparison of the Warehouse and Multidatabase Approaches" for more details), including that (a) few source DBs accept complex queries via the Internet (an immediate deal killer), (b) the user lacks control over which version of the data is queried, and over the hardware that provides query processing power, (c) the speed of the Internet limits transmission of query results, and (d) users cannot cleanse the source DBs that they query of potentially erroneous, incomplete or redundant data – that is, they cannot alter the source DBs in any way. The DB warehouse approach overcomes all of these limitations.

This paper describes an evolving open source toolkit for constructing DB warehouses that combines different collections of bioinformatics DBs within a single physical DB management system (DBMS) to facilitate queries that span multiple DBs. We emphasize that BioWarehouse is a flexible toolkit that supports multiple alternative warehouses: our goal is to enable different investigators to create different warehouse instances that combine collections of DBs relevant to their interests. The warehouse also facilitates integration of locally produced data with other public bioinformatics DBs in pursuit of goals such as capture of experimental data, sharing experimental data with collaborators, Internet publishing of data to the scientific community, and data mining and global integrative studies across multiple DBs. The data sources supported by BioWarehouse are particularly well suited to integration of pathway information, although pathways are only one of many datatypes supported by BioWarehouse.

SRI operates a publicly accessible BioWarehouse server called Publichouse that contains four of the DBs supported by BioWarehouse. See URL [Lee et al. 8] to obtain an account.

4.1.2.1.1 Motivations

The 2004 online compilation of molecular biology DBs prepared by *Nucleic Acids Research* lists approximately 580 DBs [Lee et al. 9]. These DBs represent a significant investment whose full potential has not been realized due to integration barriers. This is due to the need of different scientific projects to access information from multiple different DBs to meet their objectives. As bioinformatics DBs grow in size, and as biological questions grow in scope, the point-and-click style of DB mining becomes less and less practical in such an environment. Instead, computational biologists seek discoveries by using programs and complex queries to mine DBs. These programs seek new patterns and generalizations by issuing queries to one or more DBs to select, combine, and compute over millions of data records. Unfortunately, many barriers exist to the querying of multiple DBs, including mismatches in query language, access mechanisms, data models, and semantics.

Combining information from multiple DBs is important for two principal reasons. First, information about a given biological entity is often scattered across many different DBs: the nucleotide sequence of a gene may be stored in one DB, the three-dimensional structure of its product may be stored in a second DB, information about the expression of the gene may be stored in a third DB, and data regarding interactions of the gene product with other proteins may be stored in a fourth DB. Bioinformatics data tends to be organized around a given type of experiment (such as gene expression or protein structure determination), yet many types of scientific investigations require combining data from different experiment types. For example, one way to partially validate whether an observed protein-protein interaction is of physiological relevance is to ask whether the genes for both proteins are expressed in the same cell type; answering this question requires combining information from multiple DBs.

Second, DB integration is important because different DBs often contain redundant or overlapping information. DB integration allows cross-validation and verification of the DBs to identify such information.

In summary, advances in biology are hindered not by lack of data, but by the diversity of technologies used for storing the data. Here we propose a solution, as shown in Lee et al. Figure 1, whereby the DBs required for a project (or set of projects) are collected into a single high-performance environment operating from a local server so that scientists can control the set of

DBs, the version of each of these DBs, and the performance of the overall system. This approach also enables scientists to combine data they have produced locally with data from public bioinformatics DBs.

4.1.2.1.2 Comparison of the warehouse and multidatabase approaches

We define the multidatabase approach (also known as the federated approach) to DB integration as the approach whereby users employ a special-purpose query language to formulate a single query that spans multiple DBs [Lee et al. 10]. A mediator software system then dissects the query into subqueries relevant to individual DBs, transmits the subqueries via the Internet to each DB, combines the results of the subqueries, and returns the result to the user. Systems that employ the multidatabase approach include K2 [Lee et al. 11] developed at the University of Pennsylvania, BioKRIS [Lee et al. 12], OPM [Lee et al. 13], TAMBIS [Lee et al. 14], and BioMediator [Lee et al. 15].

This approach has two potential advantages over the warehouse approach. First, it allows users to avoid investing in the hardware required to replicate the source DBs locally. The significance of this barrier depends, of course, on the number and size of the DBs to be integrated. Second, it provides users with immediate access to the freshest version of each source DB. However, warehouses can provide fresh data if they are updated frequently. Yet, refreshing large DBs can involve frequent downloads that are costly in network bandwidth, unless the DB offers downloads of changed entries only (deltas). Very few public bioinformatics offer delta downloads, which we argue should be an area for future research, to decrease the costs of warehouse updating.

However, several limitations are associated with the mediator approach:

- A key limitation of the multidatabase approach is that it rests on a faulty assumption, namely, that all the bioinformatics DBs that users want to query are queryable via the Internet by using a network query API with the expressive power of a language such as SQL.

  That is, the assumption is that the multidatabase approach will allow us to integrate DBs that are ready and waiting in a queryable form. However, the vast majority of bioinformatics DBs have not been made queryable in this manner by their developers and this situation has changed little in the past decade. The DiscoveryLink system [Lee et al. 16] does provide technology for wrapping into an SQL-queryable from data sources that are available as files. We expect that writing such a wrapper would be similar in complexity to writing a BioWarehouse DB loader. Furthermore, once wrapped in this manner, some advantages of the multidatabase approach (such as instant access to newly released versions of the data) are lost.

- The performance of the Internet limits query speed because potentially large intermediate results must be sent across relatively slow Internet links.
- Every source DB server that is accessed by a user query must be available during execution of that query; the more sources a query accesses, the higher the probability that at least one source will be unavailable.
- The fact that users do not control the hardware on which queries submitted to a mediator are run can be problematical for users whose applications produce large numbers of queries and/or produce large query results. It is unrealistic to expect a single Internet- accessible server for a given public DB to be able to satisfy the computational demands of the entire biomedical

research community, and mediator users have no control over the hardware supporting the DBMS servers to which their queries are sent. This problem can be solved by installing a local warehouse whose hardware is configured to satisfy the query processing demands of the user's application (e.g., a large-memory multiprocessor).

- Although access to up-to-date data can be an advantage of the multidatabase approach, it can also be a disadvantage. Some users need control over what dataset version they are querying, and do not want that version changing out from under them. For example, a user who is running a software package that functions with version 3.0 of a DB, but that breaks under version 4.0 of that DB, wants control over when to migrate to version 4.0. Similarly, a user who is training and evaluating a machine learning program may want to perform experiments with respect to a constant version of the DB for consistency in evaluating program performance.
- Mediator users are unable to perform cleansing of remotely accessed DBs. In contrast, DBs loaded into a local warehouse can be cleansed of what a user considers to be erroneous or noncompliant data. If desired, this extracted data can then be stored as a distinct dataset within the warehouse.

In addition, users who generate their own local experimental or computationally derived datasets need a DBMS in which to house them. The warehouse fulfills that need, whereas the multidatabase approach is unable to store locally produced data.

Taking full advantage of the BioWarehouse and its preceding benefits does not preclude integration of databases not yet supported by the BioWarehouse. BioWarehouse may be used in a complementary fashion with a mediator system in that a multidatabase query engine can sit above BioWarehouse and enable access to additional DBs. This software can then send subqueries to BioWarehouse when it contains the relevant data, whereas other subqueries can be sent to external Internet-accessible DBs.

In summary, our intention is not to rule out use of the mediator approach, but to present what we see as a systematic discussion of the strengths and weaknesses of both approaches, many of which have not been discussed previously.

4.1.2.1.3 Definitions

A **data object** is an element of a description of a biological system at a particular granularity and with respect to a particular ontology. For example, a data object might correspond to a gene, or to a control region within a gene, or to a protein-protein interaction. A **database** (DB) is a collection of data objects. For example, Swiss-Prot is a DB. A **dataset** is a specific version of a DB. For example, Swiss-Prot version 39.0 is a dataset.

A **database management system** (DBMS) is a software system for storing and querying collections of data. A **data model** is the primitive data structuring mechanism used by a DBMS, such as the relational data model used by relational DBMSs. A **schema** is the set of all tables used to represent data objects. A **subschema** is a set of related DB tables within the warehouse schema, such as all tables involved in defining the representation of a data object.

A **BioWarehouse instance** results from integrating a specific set of datasets into the BioWarehouse schema within a single DBMS. An instance could contain multiple versions of the same DB. A **warehouse object** is the representation of a data object in a warehouse. A

warehouse identifier (WID) is an integer unique identifier assigned within a BioWarehouse instance to a warehouse object.

4.1.2.2 Implementation

4.1.2.2.1 Overall Requirements

The warehouse must scale to allow effective operation with terabytes of data. Each dataset is expected to be gigabytes in size, and available datasets typically increase significantly in size each year. **Decision:** Relational DBMS technology is in much more prevalent use in the bioinformatics community than is object-oriented DBMS technology, and has better scalability; therefore, BioWarehouse uses relational technology.

The warehouse should be compatible with standard freeware DBMSs and commercial-grade DBMSs that are most commonly used in the bioinformatics community, to allow users who cannot afford the cost of commercial DBMSs to employ freeware DBMSs, and to allow users who cannot afford the limitations of freeware DBMSs to employ commercial DBMSs. **Decision:** BioWarehouse supports the Oracle and MySQL DBMSs.

The BioWarehouse architecture should be scalable to support the integration of a growing number of data sources. **Decisions:** The project is open source to permit contributions of loaders from many groups. In addition, the complexity of the schema must be constrained to facilitate extensibility (see next section).

Multiple access mechanisms must eventually be provided including SQL, XML-based methods, Open Agent Architecture (OAA) [Lee et al. 17], CORBA, and Web-based query interfaces. **Decision:** Initial support is provided for SQL (because of its pervasiveness) and for OAA (because of its use in the DARPA Bio-SPICE program).

For installation, given an operating DBMS the requirements for the person configuring a warehouse instance are limited to basic DB administrator (DBA) expertise.

4.1.2.2.2 Schema Requirements

The BioWarehouse schema should be as simple as possible. If its schema grows too large, that complexity could be a significant barrier to widespread adoption of the warehouse technology, because users will find the schema so difficult to understand that they will be unable to write queries or application programs. Furthermore, because its schema must evolve as BioWarehouse grows to support new datatypes, a complex schema will be a significant barrier to the further development of BioWarehouse, both at SRI and by collaborators at other institutions, thus limiting the scalability of BioWarehouse to more data sources.

Decisions: We define single common tables for information that is common to many warehouse datatypes, to decrease the schema complexity. For example, comments and citations are common to many warehouse datatypes (such as to genes, proteins, and pathways), and each is implemented through a single table. We must define an association between a comment and a warehouse object. If different BioWarehouse object types used different spaces of object identifiers, uniquely defining that association would require the schema to encode both the object ID and the object type, since objects of different types could have the same ID. Thus, we use a simpler approach whereby all BioWarehouse objects share a single space of object identifiers within a warehouse instance. The identifiers are known as warehouse identifiers (WIDs), and are

integers that are assigned at DB load time. The use of a single space of WIDs allows associations between, for example, a comment and an object, to specify a WID only, and not the object type also.

The warehouse schema must support the concurrent presence, accessibility, and addressability of multiple datasets (multiple versions of a given DB) within a BioWarehouse instance.

The warehouse schema should facilitate coercion of different sources of the same type of data into a common semantic framework. For example, a warehouse might be created that contains both the Swiss-Prot and PIR protein DBs (note that UniProt does not contain all information from PIR). Once loaded, the two DBs would exist side by side within the warehouse, without having been merged, but within the common warehouse schema. A separate project within the same BioWarehouse instance could create yet a third dataset within the warehouse consisting of a nonredundant merging of Swiss-Prot and PIR. This approach implies that the warehouse user need learn only the schema of the warehouse, not of each data source, whereas some mediator systems lack a global schema and require the user to know the schema of each DB that they query.

4.1.2.2.3 Loader Requirements

Because bioinformatics DBs are often large and may have a relatively poorly defined syntax, load failures are frequently observed and without precautions could result in crashing the loader. For this reason, DB loaders should be able to recover gracefully from errors encountered during parsing of their input files. **Decision:** The loaders are designed to keep loading even in the presence of an error. If partial data has been inserted into the warehouse, a LoadError flag maintained on the related objects is updated to indicate that an error occurred while parsing the object, and that the warehouse entry for the object may therefore be incomplete or contain errors.

4.1.2.2.4 BioWarehouse Schema Design

We designed the BioWarehouse schema by first studying the schemas of each DB to be integrated, as well as the schema of other DBs that use the same datatype. Our experience in developing the Pathway Tools ontology [Lee et al. 18], which spans many warehouse datatypes, was also helpful.

The development of the BioWarehouse schema was guided by several principles, illustrated in this example involving Swiss-Prot, TrEMBL, PIR, and EcoCyc. Although the exact fields present in these DBs vary, all of them contain information about proteins; therefore, we consider the protein subsets of these DBs to be a single datatype.

Since DBs typically conceptualize proteins in different ways, any kind of cross-DB operation faces the problem of semantic heterogeneity, whereby information is partitioned in different fields that use different definitions (such as different units of measure). One possible approach to supporting protein DBs within the warehouse would be to create different schema definitions for each of the conceptualizations of proteins used by the source DBs. However, this approach would perpetuate the semantic heterogeneity among these DBs, and would complicate the resulting schema of BioWarehouse. At the extreme, BioWarehouse would have to contain a different protein schema for every protein DB that it loads. Therefore, the warehouse schema would be larger and more complex, and users would have more difficulty understanding the schema, making it more difficult for a user to query the DB. Under this approach, a user who

wanted to query all proteins in the DB would have to study the subschema for each different protein DB in the warehouse, and essentially write a separate subquery for each subschema.

Instead, the warehouse approach uses a single set of schema definitions to cover a given datatype, even if that datatype is conceptualized differently in different DBs. For example, we create a single set of schema definitions to span all attributes for proteins. The DB loaders are responsible for translating from the conceptualization used in each DB within the family to the conceptualization used by the BioWarehouse schema. This approach eliminates the semantic heterogeneity of these DBs, allowing users to query all protein sequence DBs using the same schema.

Another important element of our approach is to explicitly encode the dataset from which each data object within the warehouse is derived. For example, since entries from any protein DBs are loaded into the same set of tables, it is critical for user queries to be able to distinguish Swiss-Prot entries from TrEMBL entries. Thus, queries can request the retrieval of all warehouse protein sequences that were loaded from the Swiss-Prot dataset.

Our DB loaders do not attempt to remove redundancy during the loading of multiple DBs, so if Swiss-Prot and PIR were loaded, and contained entries describing the same protein, two distinct entries would be created in the warehouse. This is important for four reasons: (1) Scientifically, we believe that the warehouse should respect and maintain the integrity of individual datasets, that is, the warehouse should preserve information about the source (provenance) of warehouse entries so that users can determine exactly what proteins are part of the Swiss-Prot dataset or the PIR dataset. (2) Swiss-Prot and PIR might provide different information about the same protein, either because they disagree about the biological facts, or because they provide different commentary. (3) It simplifies the loaders, which need not be concerned with detecting and removing redundancy between different datasets. (4) It allows later execution of algorithms for computing nonredundant protein datasets, where different algorithms can be appropriate for different applications. A nonredundant protein DB could be created as a separate dataset within the warehouse that resulted from applying a redundancy-reduction algorithm to the Swiss-Prot and PIR datasets. This approach satisfies those users who will want to study Swiss-Prot per se, and those users who will want to work with a large nonredundant protein DB.

Each dataset is described by a row within the DataSet table, and contains attributes such as the dataset name, version number, release date, and reference URLs.

Our schema design also allows multiple versions of a given dataset to be loaded simultaneously, thus supporting queries that study the relationships among different versions of a dataset (such as: Is the error rate of Swiss-Prot increasing or decreasing? At what rate is the number of ORFs in the bacterial genomes within Swiss-Prot decreasing over time?). Having access to multiple versions of a DB simultaneously is also important when a newer DB version has changed in a way that interferes with important application software, or when one user is in the middle of a computational study that he wants to complete on the older DB version to maintain a consistent set of results, and another user of the same BioWarehouse instance wants to access the newest version of Swiss-Prot. Our approach gives warehouse administrators the option of deleting the old version of a DB and loading its new version, or loading the new version alongside the older one, allowing both to exist simultaneously.

This ability to maintain distinct datasets also enables users to capture locally produced data (e.g., protein-sequence data) in the warehouse by defining a separate warehouse dataset to hold these data.

Consider the fact that proteins, pathways, and other bioinformatics datatypes all need to cite literature references. We do not want the protein and pathway datatypes within BioWarehouse to refer to different schema definitions of literature references, as would be done in warehousing approaches that create separate data-source-specific subschemas for every data source they include. We also do not want them to contain duplicate copies of the same definitions of literature references, which can happen with normalization schemes that blindly create new tables for every multivalued attribute of an entity. Therefore, we encode citations from all BioWarehouse datatypes and datasets within a single citation subschema within BioWarehouse. DB links are treated in a similar fashion – the subschema for each warehouse datatype encodes links to other DB objects in the same way.

Note that the current version of the warehouse schema is more oriented toward prokaryotes than eukaryotes; over time better support for eukaryotes is being added. This orientation applies to representation of genes, for example, the schema does not currently represent introns or alternative splicing.

4.1.2.2.5 BioWarehouse Schema Implementation

BioWarehouse supports several bioinformatics datatypes, each of which is implemented as one or more tables in the schema. The full schema is provided as supplementary material to this report as an entity-relationship (ER) diagram [see Lee et al. Additional file 1] to provide an overview of its organization, and as an SQL definition for readers interested in its details [see Lee et al. Additional file 2]. Lee et al. Figure 2 shows the main datatypes in the BioWarehouse schema, and the relationships between them. From left to right within this figure:

- **Taxon:** Describes taxonomic groups such as genus or species. Taxa are related to one another to represent the various hierarchical levels of taxonomic classification.
- **BioSource:** Represents the source of a biological material, whether nucleic acid molecule, protein, or gene. In addition to specifying the taxon of the material by cross-referencing to entries in the Taxon table, BioSource also stores source descriptors such as cell type, tissue, and sex of the organism. The loaders provided with the BioWarehouse all reference the NCBI Taxonomic DB [Lee et al. 19] when creating entries in BioSource to provide a normalized set of taxonomic classifications.
- **Nucleic Acid:** Defines a DNA or RNA molecule, or a fragment thereof. A single contiguously sequenced fragment of a larger nucleic acid is represented by an entry in table SubSequence.
- **Gene:** BioWarehouse uses the prokaryotic notion of gene – a region of DNA that codes for a protein or RNA product, beginning with the transcriptional start site. Genes are related to entries in the NucleicAcid and Subsequence tables that define their sequence, and to their protein products. A gene may also be directly related to a BioSource.
- **Protein:** Defines proteins, including their amino acid sequences. Proteins are related to the genes that encode them, to the reactions they catalyze, and to features defined on their sequence. A protein may also be directly related to a BioSource. The schema can capture the subunit composition of a multimer, and does not require that a protein sequence be present.

- **Feature:** A subsequence of interest on an amino acid or nucleic acid sequence, such as a catalytic site, a phosphorylation site, a promoter region, or a transcription-factor binding site.
- **Reaction:** A spontaneous or catalyzed chemical reaction. A reaction is related to the protein that catalyzes it; to pathways that contain it; and to its chemical substrates, which can be small molecules or macromolecules such as proteins.
- **Chemical:** A small-molecular-weight chemical compound.
- **Pathway:** An ordered collection of biochemical reactions.

The bottom of Figure 3 lists some additional tables within the schema. Every entity within the warehouse (e.g., every nucleic acid, protein, and gene) has a row in the Entry table that defines metadata such as the time it was inserted in the warehouse, and its time of last update. Every warehouse entity is also associated with the dataset (DB version) from which it was derived; datasets themselves are defined in the DataSet table.

The Term table implements storage of external controlled vocabularies such as those provided by an ontology. Each entry in the Term table is a single term in a controlled vocabulary. The Enumeration table implements controlled vocabularies internal to BioWarehouse itself. For example, the BioWarehouse table BioSource contains a column "Sex" that is allowed to take on several possible controlled values including "Male" and "Female." The Enumeration table specifies the allowed controlled values for specified BioWarehouse tables and columns, making BioWarehouse self-documenting.

In addition to the preceding bioinformatics datatypes, the BioWarehouse represents many relationships among these datatypes. If the relationship is one-to-one, a column in the corresponding table simply contains the WID of the associated object. For example, a gene is associated with its nucleic acid in this manner. Relationships of higher multiplicity, such as many-to-many relationships, are implemented as linking tables that associate two or more primitives, such as a gene and a protein. A linking table contains WIDs for each of the related objects. This permits efficient, normalized representation of these relationships.

The warehouse also contains tables that implement auxiliary information, including descriptions of the source dataset of each warehouse entry, literature citations, human- and software-generated comments, DB cross-references (both internal and external to the warehouse), synonyms for named objects, and controlled vocabularies.

The BioWarehouse schema will evolve as new loaders are added to support new datatypes. SRI encourages other groups to submit new loader implementations, Java library extensions, and schema extensions to this open-source project. We do feel it is necessary for SRI to be the final arbiter of such extensions to ensure that the schema implementation remains compatible with its design principles to ensure continued maintainability and scalability of BioWarehouse. New versions of the schema will sometimes be incompatible with old versions, which will require loader modifications, and data reloading. Since public data sources must in any event be reloaded at regular intervals, the requirement of data reloading is not a large burden.

To support multiple DBMSs and their different flavors of SQL, each DBMS-specific schema is generated from a common schema template. The schema template consists of a framework using SQL syntax that is common among all supported DBMSs, interspersed with variables. Macro

substitution converts the common schema to one that is conformant to the DBMS. The most common substitution is for primitive data types, which differ significantly across DBMSs.

4.1.2.2.6 DB Loaders

It is the responsibility of each loader to translate the flat file representation of its source DB into the warehouse schema. Typically, many of the source DB attributes are copied into the warehouse either verbatim or with minor transformations (e.g., converting "YES" and "NO" to "T" and "F"). The few source attributes not represented in the warehouse are generally ignored, although some attributes are added to the warehouse CommentTable. An example of warehouse translation semantics is shown in Lee et al. Table 1 for one file from the BioCyc collection of DBs.

Loaders have been implemented in both the C and Java languages. C-based MySQL loaders interface with MySQL using the C API provided as part of MySQL. C-based Oracle loaders interface with Oracle using the Oracle Pro-C precompiler. Java-based loaders use the Java Database Connectivity (JDBC) API to interface with the DBMS. Each of these APIs allows SQL to be embedded and/or generated within its source language.

Loaders have been implemented for these bioinformatics DBs: UniProt (Swiss-Prot and TrEMBL [Lee et al. 20]), ENZYME [Lee et al. 21], Kyoto Encyclopedia of Genes and Genomes (KEGG) [Lee et al. 22], the BioCyc collection of pathway/genome DBs (see URL [Lee et al. 23-25]), the NCBI Taxonomy DB [Lee et al. 19], GenBank [Lee et al. 26], the Comprehensive Microbial Resource (CMR) [Lee et al. 27], and Gene Ontology [Lee et al. 28].

The architecture required to load datasets into a warehouse instance is akin to the process of compilation, but with the source code being a dataset and the object code being SQL insert statements to add the contents of the dataset to the warehouse. Standard parser generation tools (ANTLR for Java, Flex and Bison for C) are used throughout to specify the syntax of the input files, and associated loader actions.

A set of support routines is provided to create and manipulate an internal representation of the warehouse objects, including assignment of WIDs to objects, and the resolution of intra-dataset cross-references into WIDs. Loaders may make SQL queries to the local warehouse. SQL generation is performed by translating the internal representation into SQL statements.

Each loader has an associated manual describing its operations and any limitations in the loader. For example, the GenBank loader is currently recommended for use on prokaryotic sequences only.

4.1.2.2.7 BioWarehouse Java Utility Classes

The BioWarehouse implementation provides a set of Java classes with general utilities for interacting with BioWarehouse. These classes are useful for developers who want to construct new BioWarehouse loaders or applications. The classes provide methods for connecting to the database and for loading data into a BioWarehouse instance. These classes are packaged as a single module (a Java JAR file) that can be used by any Java-based loader or application. To abstract underlying DB specifics from the developer, a client application uses a factory class to obtain an instance of a Java DB class (Oracle or MySQL). The DB class provides methods for connecting to the database, performing queries, or doing BioWarehouse-specific tasks, such as obtaining a WID in an object-oriented, DBMS-independent way.

Similarly, we provide Java classes enabling a client to interact with the BioWarehouse tables in an object-oriented manner. The table classes are defined by an extensible class hierarchy that defines one class per table. The hierarchy greatly simplifies implementation of new table classes. Similar table types (e.g., all object tables or all linking tables) have their common functionality factored out into a base class. Each class provides accessors with methods for each property (column) of the table. These classes enable developers to create instances of the table classes (representing a single row in the table), set properties of the table object (corresponding to inserting values into the table), and store the data into the database. Methods are also provided to retrieve a row from a table given its primary key(s) (e.g., given a WID for object table entries), and to update or delete that row. Lee et al. Table 2 compares the processes for the SQL versus the Java method. In these examples, the Protein.DataSetWID is 2 and the Protein. WID is 5.

In addition to providing a convenient object-oriented interface to the BioWarehouse schema, the Java schema classes leverage the metadata capabilities of JDBC to perform data checking before inserting data. For example, we can detect if a text length exceeds the allowed maximum for a column before attempting an insert (which would otherwise fail). Another benefit is that we can perform unit tests on the classes to ensure compatibility with the current implementation of the BioWarehouse schema, and to confirm that all functions work as expected with each type of DBMS.

4.1.2.2.8 Publichouse: Publicly Queryable BioWarehouse Server

As a convenience to users who do not want to maintain their own local BioWarehouse instance, SRI provides a publicly queryable BioWarehouse instance called Publichouse, which stores the open BioCyc, NCBI Taxonomy, ENZYME, and CMR DBs. Users can query Publichouse via Internet SQL queries. Because Publichouse stores only those DBs that their creators make openly available, users wanting to query other BioWarehouse-supported DBs must load those DBs into their own local BioWarehouse instance. That is, SRI cannot typically redistribute DBs that are not openly available, but most users will be able to download and install those DBs for their own local use.

Currently, Publichouse contains BioCyc DBs for ten organisms. However, in the near future we expect that number to increase to more than 150 organisms due to a joint effort between SRI and the Computational Genomics Group at the European Bioinformatics Institute to generate Pathway/Genome DBs for every completely sequenced bacterial and eukaryotic genome.

4.1.2.2.9 User Support and Documentation

Extensive documentation is available for BioWarehouse within the software distribution. The available documentation is listed in a table of contents within the distribution at http://biowarehouse.ai.sri.com/repos/doc/index.html. The BioWarehouse documentation set includes release notes, a quick start guide, environment setup documentation, schema description and DBMS setup instructions, a description of the integration with the Dashboard for the February 2004 Bio-SPICE demonstration, and descriptions of Perl utilities and Perl demo scripts.

The table of contents also has a table listing statistics about each loader (latest supported version of its DB, last input DB size, load time, etc.) For each loader, there are two pieces of documentation: how to build and run the loader, and a manual for developers describing the details of the loader implementation and mappings from the source DB schema to the BioWarehouse schema.

Bug reports and requests for assistance should be sent to support@biowarehouse.org.

4.1.3 The MATLAB Bridge and Other SRI Developed Tools

4.1.3.1 The BioMat Bridge

The current version of the BioMat Bridge is 1.2.2. The Bridge is a polymorphic workflow object that integrates MATLAB ™ into the Bio-SPICE Dashboard. The requirement for this analyzer was articulated by a number of community members and reflects that fact that MATLAB is widely used in the larger biology and bioinformatics communities. Its creation required a significant effort, but it permits the leveraging of existing and legacy MATLAB code within the Dashboard

The BioMat Bridge enables a workflow in the Bio-SPICE Dashboard to enter (and exit) MATLAB 's computation engine. It allows MATLAB functions to seamlessly interact with workflow objects in the Dashboard without any restrictions on what the MATLAB code actually does. Depending on how it is configured, a BioMat Bridge object may perform analysis, visualization, simulations, collect user input, generate data, etc. Anything that can be done with MATLAB can be integrated into a workflow using the BioMat Bridge. As such, the BioMat Bridge provides more generic functionality than other workflow objects.

Three important concepts underlie the BioMat Bridge, polymorphism, processing function and mapping.

4.1.3.1.1 Polymorphism

Unlike other workflow objects, the BioMat Bridge is polymorphic. Depending on how it is configured, the BioMat Bridge can take on any number of distinct forms and functions.

Among other things, the configuration defines the number and data types of any input and output parameters and what MATLAB functionality to invoke. The interface to the workflow and functionality are undefined until the BioMat Bridge is configured.

4.1.3.1.2 Processing Function and Parameters

The BioMat Bridge invokes MATLAB through a single function call. An object's configuration specifies the particular function that will be evaluated. We call this the processing function. Of course, the processing function can call other functions as well. From inside the processing function, you are free (and encouraged) to take advantage of anything MATLAB has to offer.

Typically, the processing function is an m-file or a p-file on the local filesystem. However, the BioMat Bridge also supports the execution of a processing function that is supplied as an m-file input from the workflow. For example, the contents of an m-file could be generated on-the-fly during the execution of a workflow and passed to a BioMat Bridge object to be used as the processing function. Note that the signature of the input supplied processing function must still be known and provided in the configuration.

Now, how do we provide a processing function with the required input and collect its output?

This is accomplished by automatically associating each of the processing function's calling arguments and each its return values with a workflow parameter. Parameters are what make up the external interface for all workflow objects. All parameters are classified as either input

parameters or output parameters. Workflow objects receive their input through their input parameters and pass their output along through their output parameters.

In a configured BioMat Bridge object there will be an input parameter to supply a value to each non-reference calling argument in the processing function. Each reference calling argument and each return value will supply an output parameter. The parameters take the same name as their associated argument or return value, respectively. In the case of an input supplied processing function, there will be an additional input parameter for the m-file data.

You can see that there is a one-to-one relationship between the workflow parameters (ignoring the case of an input supplied processing function) and the calling arguments and returns values of the processing function. The signature of the processing function defines the object's interface to the workflow. With that in mind, if you wish to create a BioMat Bridge object with a specific workflow interface, you may need to use a wrapper function with the appropriate signature as your processing function.

4.1.3.1.3 Mapping

One of the challenges of interacting with MATLAB is how to make data from a workflow available for use within a MATLAB environment. Similarly, we need a way to make data produced in a MATLAB environment available for use in a workflow.

Inside the Bio-SPICE Dashboard, workflow objects interact and exchange data through their input and output parameters. Each parameter is associated with a Java data type that is used to convey the data. Within a BioMat Bridge object, each parameter corresponds to either a calling argument or a return value of the processing function. Therefore, in order to interact with MATLAB through the processing function the BioMat Bridge must know how to convert data back and forth between Java and MATLAB representations.

We use the term "mapping" to describe a specific method for performing such a conversion.

The BioMat Bridge supports a variety of different mappings for converting between the Java types used in the Dashboard and the most common MATLAB types.

4.1.3.1.4 Example Workflows

4.1.3.1.4.1 Calling a Simple Function

This simple example illustrates how to configure the processing function and giving it an argument of a primitive type. Follow these steps.

Save the example function myfunc.m and the XML configuration file config-myfunc.xml to the local hard disk. These files can be found in the Bio-SPICE code repository with the BioMat Bridge documentation module.

Place a BioMat Bridge in the workflow and access the context menu (right-click) to select 'Edit -> Analyzer Configuration' to open the configuration dialog window. Click on the 'Load...' button at the lower left corner to load the configuration file config-myfunc.xml that you saved in the first step. Note the change of the window title bar: it displays the file name of the configuration file followed by the word '(invalid)' to indicate an incomplete configuration.

Switch to the tab 'Search Paths' by clicking on the tab or pressing <ALT>-P. Under the list of search paths should be a red line saying YOUR PATH TO myfunc.m. Select this line, click on

the 'Delete' button, and use the 'Browse & Add...' button to add the directory, in which you had saved the file myfunc.m from step one.

Next, click on the tab 'Default Values' or press <ALT>-D to enter a value for the input parameter `userName`. To do so, click into the rightmost field of the line corresponding to the input parameter (in this example, there is only one line). Enter a name of your choice and hit 'Return' or click somewhere else to finish editing the name.

Optionally, you can save the modified configuration using the 'Save As...' button to preserve your changes for the future.

Close the configuration dialog with the 'OK' button. If you have not saved the modified configuration, a warning message should appear asking you whether to proceed.

Back in the workflow, the bottom of the analyzer should now turn green signaling that we are ready to go. Start the workflow and be patient as the connection to the MATLAB engine takes a while to establish. Another output window will appear that repeats all text output from the MATLAB engine similar to Figure 17:

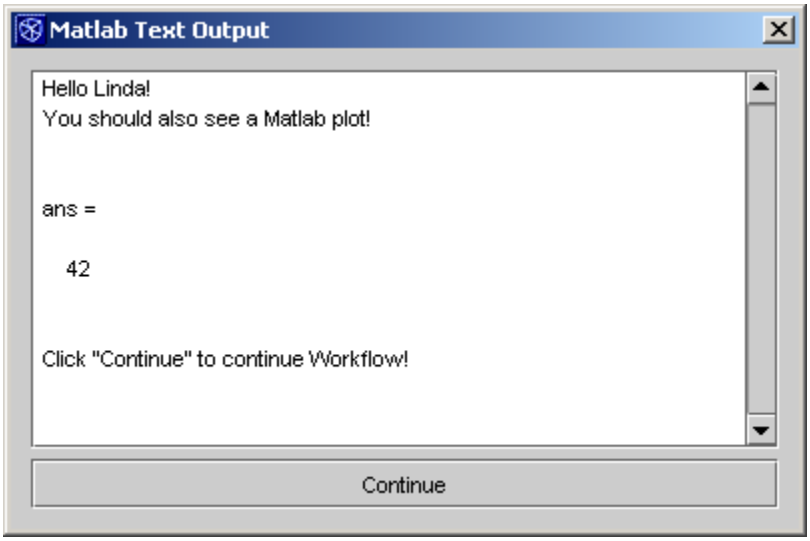

Figure 17. BioMat Bridge Example MATLAB Text Output.

In the configuration, we have set the option `suppressOutput` to be unchecked (consult the tab 'Options' in the configuration dialog). Then, the processing function is called without a trailing semicolon in MATLAB. Therefore, the MATLAB text output also shows the value of the return values, here "r = 42." You can use this feature for additional error checking. Note that we do not configure the analyzer to use this return value for an output connector.

Through the configuration of the analyzer we have also instructed this window to stay open when MATLAB is finished. The 'Continue' button will then be enabled and the user clicks this button to instruct the BioMat Bridge to continue the workflow.

4.1.3.1.4.2 Toxicology Use Case (originally developed Aug '04)

This example uses many of the new native mappings between data types.

Save the processing function process_tox_data.m , the example input data ToxicologyData.txt , and the XML configuration file config-tox.xml to the local hard disk. These files are located in the BioMat Bridge which can be found by clicking the Help Contents button under the Dashboard's Help pull down menu.

Place a BioMat Bridge in the workflow and load the configuration file you had just saved using the configuration dialog. In the 'Search Paths' tab, delete the red line reading YOUR PATH TO DIRECTORY CONTAINING process_tox_data.m. Use the 'Browse & Add...' button to add the respective directory to the list of search paths. Optionally, you can save the modified configuration using the 'Save As...' button to preserve your changes for the future. Close the configuration dialog with the 'OK' button. If you have not saved the modified configuration, a warning message should appear asking you whether to proceed.

When right-clicking the configured BioMat Bridge and selecting 'Edit -> Input Parameters' now, it should show the input configuration like Figure 18. Review the default settings (note that you cannot edit the default settings in this dialog; go back to the configuration dialog to do so).

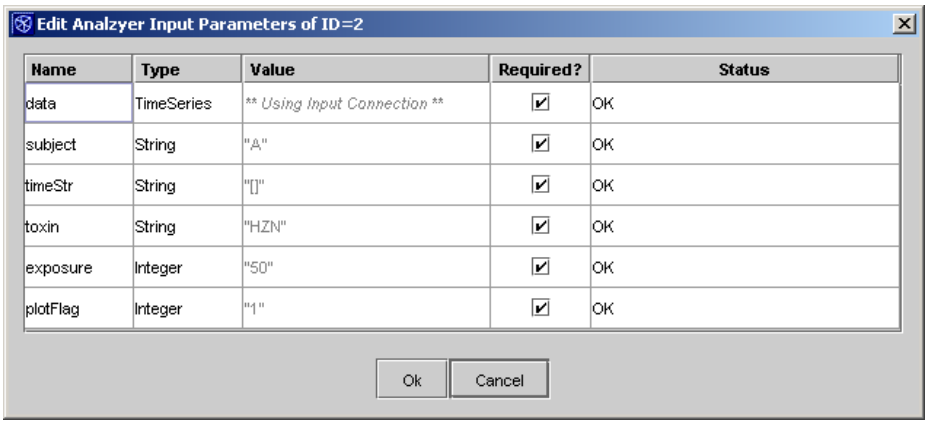

Figure 18. BioMat Bridge Input Parameter Edit Pane.

Setup a workflow according to the following steps. This screenshot in Figure 19 shows the complete workflow.

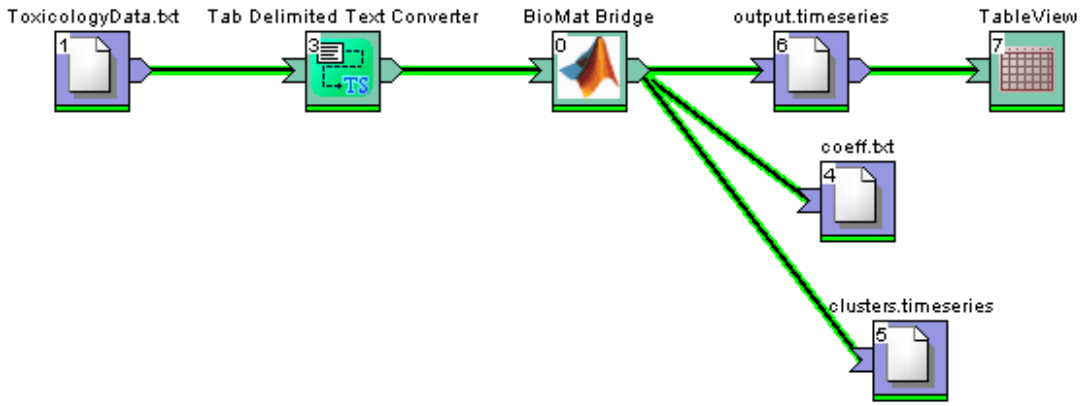

Figure 19. BioMat Bridge Example Workflow.

- Add one source document that points to the raw toxicology data in the file ToxicologyData.txt that you had save during step 1.
- Place a Tab Delimited Text Converter analyzer between the source document and the BioMat Bridge. Connect its input to the output of the source document. A dialog to edit the parameter mappings on this connector will appear because there is more than one match of parameters. Add the mapping between DocumentOutput : TextFile to dataFile : Text File by selecting the respective rows in the tables with unused parameters and clicking the button 'Add Mapping' as seen in the screen shot in Figure 20.

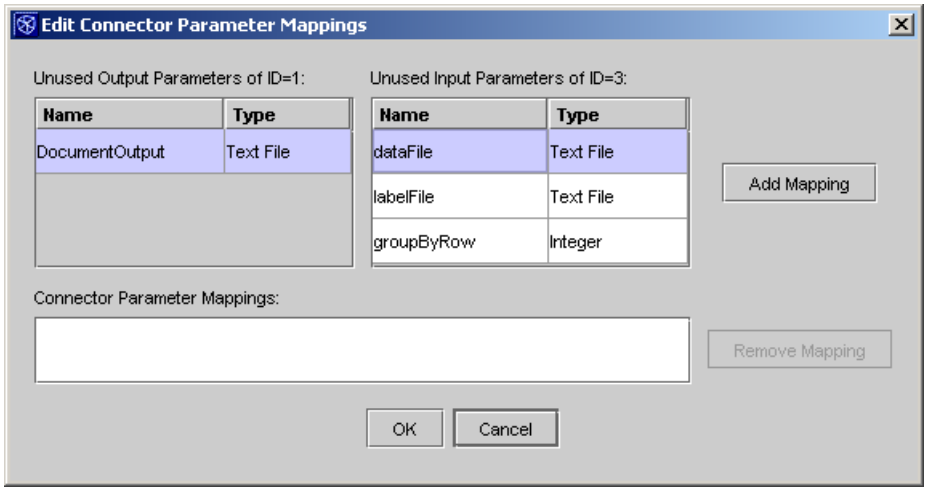

Figure 20. BioMat Bridge Edit Connector Parameter Mappings.

Then, connect the Tab Delimited Text Converter analyzer to the input of the BioMat Bridge.

- Add two destination documents that you can point to filenames coeff.txt and clusters.timeseries and connect them to the output of the BioMat Bridge. Both times, the parameter mapping editor window will appear. Map coeff : Double to DocumentInput : Object for the connection between the BioMat Bridge and the destination document named coeff.txt. Map clusters : TimeSeries to DocumentInput : Object for the connection between the BioMat Bridge and the destination document named clusters.timeseries. Figures 21 and 22 show the respective screenshots that result from these two mappings in the editor dialog.

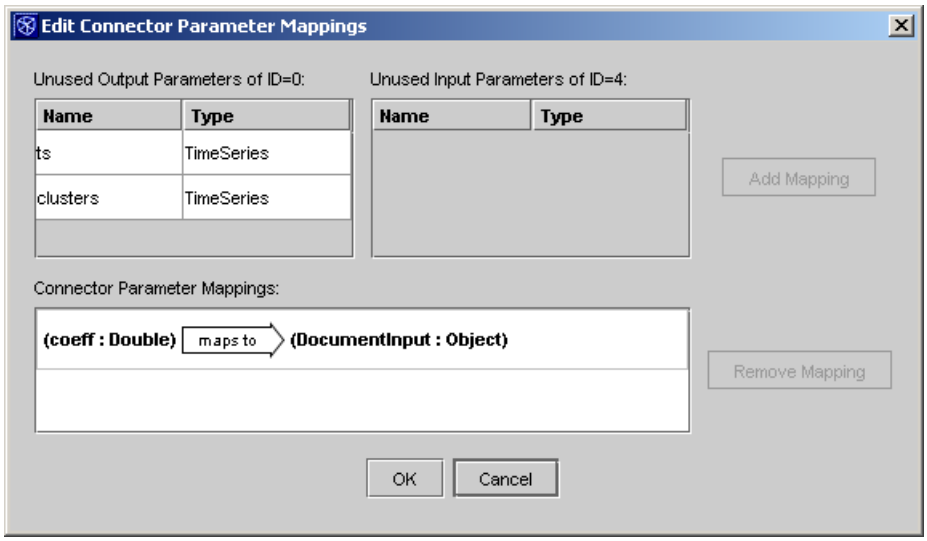

Figure 21. BioMat Bridge Results of Parameter Mapping 1.

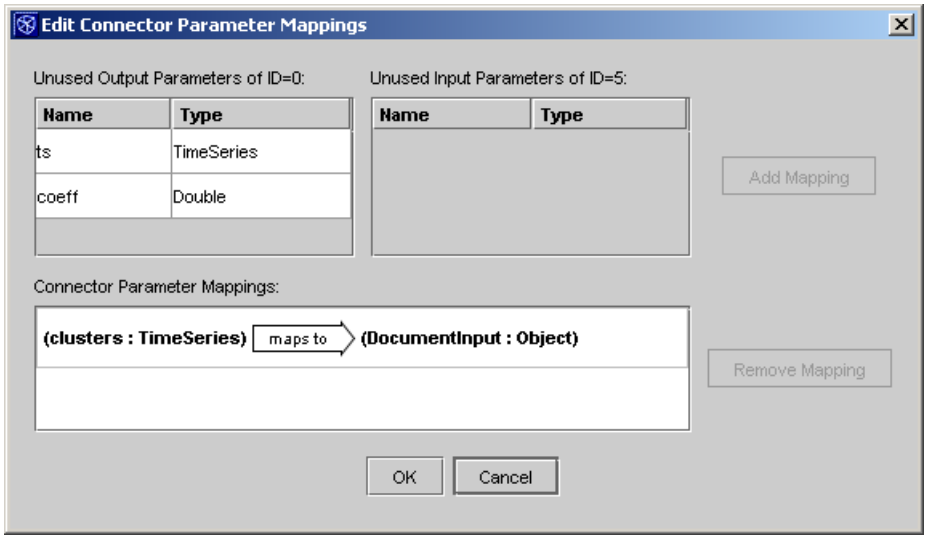

Figure 22. BioMat Bridge Results of Parameter Mapping 2.

– Add an intermediate document that you can point to filenames output.timeseries and connect it to the output of the BioMat Bridge. Again, the parameter mapping editor window will appear, in which you add the mapping from ts : TimeSeries to DocumentInput : Object. See the screenshot in Figure 23.

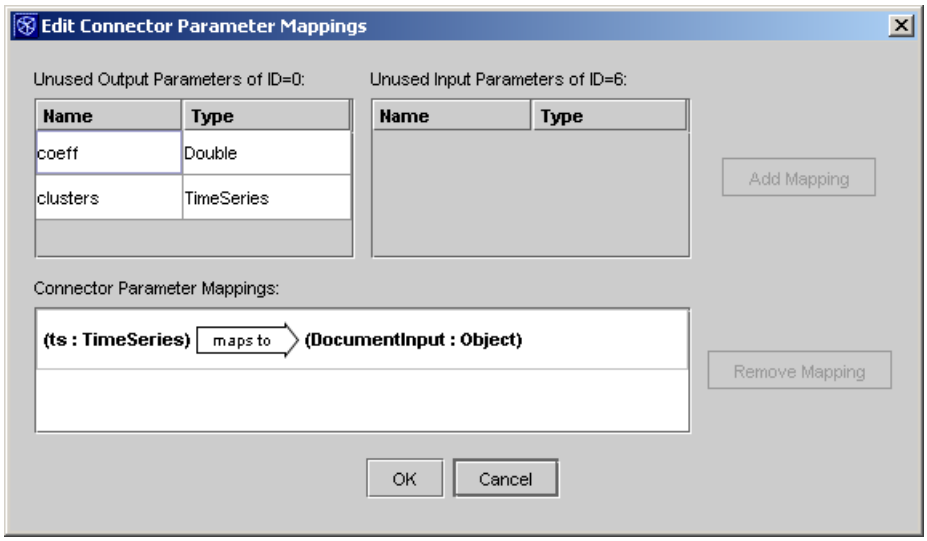

Figure 23. BioMat Bridge Parameter Mapping an Intermediate Document.

– Finally, attach a viewer such as the Table View analyzer to the output of the intermediate document. If the parameter mapping editor window shows up for this connection, choose ts : TimeSeries as the output parameter to be mapped to this analyzer's input.

Run the workflow and observe how MATLAB produces a dendogram figure like that illustrated in Figure 24 with the results of the clustering process performed.

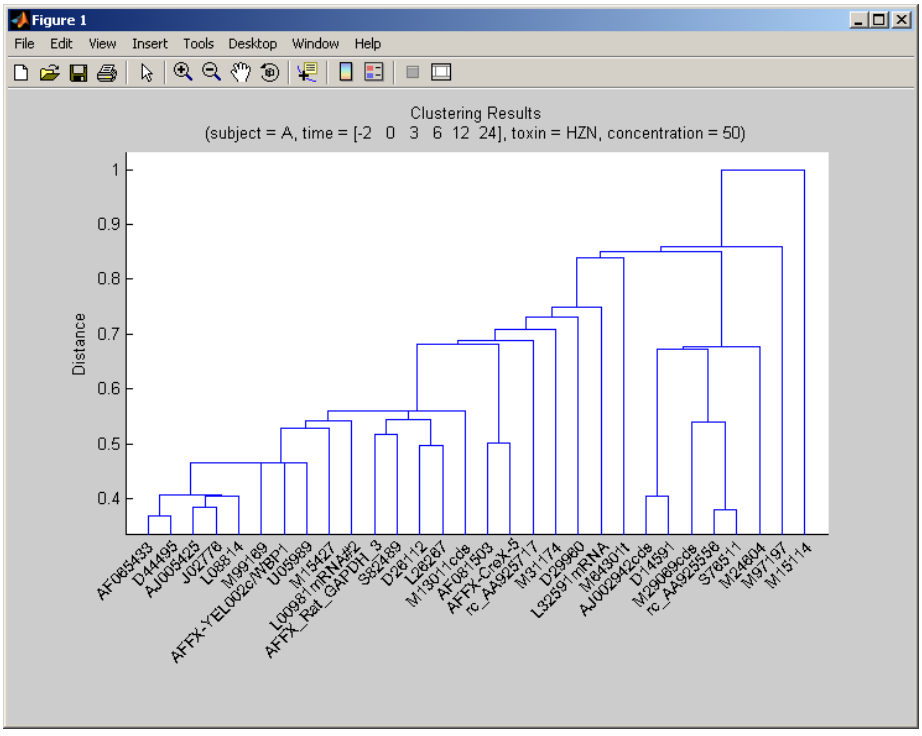

Figure 24. BioMat Bridge Dendogram Figure-Output.

4.1.3.1.5 DBAgent

The Bio-SPICE DB Agent provides an OAA query service for relational databases. The design of the agent was inspired by the Java Database Connectivity (JDBC) API and provides similar functionality for OAA users. It was developed to support the Bio-SPICE Data Warehouse effort at SRI but it is of generic utility and can be used to access data of any kind.

The current Bio-SPICE DB Agent implementation has been tested using Oracle 8.1.7, PostgreSQL 7.2.1, and MySQL Ver 11.17 Distrib 3.23.49a. A properties file is used to set the JDBC driver and connection URL at runtime so the DBAgent should work with any database for which there is an available JDBC driver.

The agent itself is written in Java but like all OAA agents it can service requests from clients written in any language supported by OAA.

Complete source code is provided for the DB Agent as well as extensive API documentation. A complete example client implementation is included that demonstrates the usage of the entire DB Agent API.

4.1.3.1.6 ptPlot Module

ptPlot is a 2D data plotter and histogram tool implemented in Java. Ptplot can be used as a standalone applet or application, or it can be embedded in your own applet or application. ptPlot was developed at the University of California, Berkeley. This module wraps ptPlot as a Dashboard accessible resource.

4.1.4 Abstraction, Anthracis, and Pathways

The broad set of tools brought to bear by the entire Bio-SPICE community, integrated and communicating via Bio-SPICE-standard languages and interfaces were put to the test in several biologically-relevant use case. In some of these use cases, the existing tools and approaches were found wanting.

A significant barrier to application of mainline Bio-SPICE simulation tools is the development of accurate and complete numeric models. For example, even in the relatively modest pathways employed by small bacteria such as *E. coli* and *B. anthracis* there are over a thousand relevant proteins and small molecules, and other a thousand reactions to consider. Since each molecule or complex could potentially effect each reaction, we have over a thousand times a thousand, or over one million unknown quantities effecting the simulation, even for just first order effects. Second-order and higher-order effects (such as the concentration of some other small molecule affecting the catalytic effect of a catalyst on a specific substrate) lead to exponentially more unknowns in the fully detailed model. The catalytic effect of certain proteins or other complexes on reaction rates has been qualitatively defined for many large pathways, but the quantitative measurement of reaction rates and effects of catalysts and other environmental factors in the cell remain largely unknown in the scientific literature. Tools have been developed by Tyson's lab at Virginia Tech [24] and Doyle's University of California at Santa Barbara lab [25] that are usable through Bio-SPICE enabling fitting of system parameters to observed quantitative data. However, for many present problems of biological interest, the amount of quantified data is small (in the tens of experiments) while the number of unknowns is large (in the thousands to millions). Thus we seek alternate methods to analyze complex biological systems until high-

throughput biological data gathering techniques are developed to populate our biological data warehouse with sufficient data to reliable fit a model.

Together with other Bio-SPICE research teams, SRI International developed such symbolic or pre-numeric analysis techniques to enable biologists to determine certain aspects of their complex biological systems even before a single rate constant is known for certain in the key pathway. Our overall research approach focused on abstraction: taking a problem so complex it is beyond the scope of current techniques, and simplifying it while preserving a property of interest. For example, we may be able to abstract a system such as bacterial metabolism to preserve properties intrinsic to equilibrium flow of chemical species through the pathway. Such an abstraction may be useful to predict information about certain questions of interest, though it would be useless in the search for timing information, for example. This type of property-sensitive abstraction of complex system descriptions may enable biologists of the future to analyze systems as complex as the human immune system from one aspect at a time. For each aspect, our symbolic and hybrid abstraction techniques provide means to simplify complex problems to the point that brute-force search and exhaustive symbolic analysis become feasible. Further, our principled techniques enable one to translate analysis in the simplified or abstracted domain back into the actual (more complex) concrete domain of discourse.

For example, if a biologist addresses a question of stability of a complex system, we might provide a hybrid abstraction of that system suitable for exhaustive symbolic analysis with a model checker. In the abstracted domain, our model checker may find a counterexample to the hypothesized stability (a trajectory of points in the abstract space that continues indefinitely without reaching a stable equilibrium). Using the techniques we propose, we can reverse the mapping, and concretize the example trajectory back into the language of the user, showing a nonstable trajectory through the original complex system. The biologist may then ask about other properties (perhaps time to quiescence) that require completely different abstraction (this one sensitive to the time constants, while the stability question could potentially be addressed in a time-free model). And after that, the biologist may be concerned with yet another abstraction of the system. In this way, a biologist may be able to rapidly understand various aspects of the complex system of interest.

4.1.4.1 Abstraction and Hybrid Systems Modeling of Biological Systems

Working together with other Bio-SPICE performers, SRI International developed methods tools, and techniques to enable simulation and analysis of discrete, continuous, and hybrid models of biological pathways.

The main goal of this task was to analyze models of biological phenomenon to improve their understandability by practicing biologists. We explored various modeling formalisms and different kinds of analyses for these formalisms to achieve this goal. Full details of the work can be found in relevant publications referenced below. We used three different modeling formalisms: continuous dynamical systems, discrete state transition systems, and hybrid systems that are obtained by combining the two formalisms. Here we give an outline of the methods used and results obtained. Key results here are joint with other authors, including other Bio-SPICE performers, other affiliated researchers, and students.

In collaboration with Ronojoy Ghosh and Claire Tomlin (both then of Stanford University), we modeled the Delta-Notch intercellular lateral signaling mechanism using hybrid systems [26].

The hybrid modeling approach can be used to eliminate nonlinearities and replace them by discrete switches. This reduces the number of continuous dimensions of the original model and opens the possibility of using new techniques for analyzing the otherwise inaccessible nonlinear models. Using a hybrid model of the Delta-Notch lateral inhibition mechanism, we showed that for a large range of parameter values, each cell has two distinct stable states: high Delta and low Notch, or low Delta and high Notch. Moreover, if a cell stabilizes in one of these two states, then it influences the stable state of its neighbors in such a way that no two neighboring cells both have high Delta. This is an example from a large class of biological systems that exhibit the interesting property of developing from a homogenous sheet of cells into a patterned sheet of cells. The analysis was done using symbolic techniques for creating abstractions of hybrid systems and model-checking them. The important feature here was that the analysis was done for a generic set of parameter values, and not for any specific value. The symbolic analysis was then compared with standard continuous modeling techniques and biological observations in the literature. The overall characteristics of the analysis were born out. This work helped show that the hybrid model of the Delta-Notch mechanism is robust to changes in the parameter values, and was reported at Bio-SPICE PI meetings. Further details can be found in [26].

We also developed a hybrid model of the sporulation initiation network *of Bacillus subtilis* by combining the cell signaling pathways that collect information from external stress signals with downstream gene regulation. We chose to model *Bacillus subtilis* since it is a close relative to the biowarfare agent *Bacillus anthracis*, yet it is safe to humans, and relatively well-studied. Our model of *Bacillus subtilis* stress response was developed in collaboration with Adam Arkin and Denise Wolf of Lawrence Berkeley Labs. The key feature of this model was that it was built using small components, where each component performed some human-understandable function. Each component was modeled as a hybrid system. The interaction of the various components contributes to the final decision the cell takes on sporulation. The use of components was important both from an automated analysis perspective, as well as from the perspective of human understanding of the roles of various proteins in the global decision to sporulate. The lack of concrete parameter values in the scientific literature meant that we could only perform the analysis in symbolic mode. However, since the number of parameters was large, we had to use additional constraints on the parameters to successfully verify the bistability of the model. The validation of these constraints needs to be done via experiments, since the constraints relate the different rate constants of reactions that occur in the sporulation network. As one example result of our symbolic analyses of *Bacillus subtilis* sporulation, we predicted that if two specific rate constants were equal, under no values of other rate constants could the system be bistable. Since the natural system is observed to be bistable, this implies that the two rate constants in question should be far apart. This type of qualitative prediction from symbolic analysis is unreachable with traditional analysis methods that rely on accurate rate constant information (usually gleaned from prior scientific literature) in order to drive a continuous simulation. Additional details can be found in [27].

Another example problem we studied was at the opposite end of the system complexity scale and involved blood glucose and insulin levels. Blood glucose concentration is maintained within strict bounds in healthy humans. Using a continuous dynamical system approach to organism-level modeling of blood glucose metabolism, we computed the lower and upper bounds of glucose concentrations in (models of) Type 1 diabetic patients. The crucial observation here is that when there is a spike in blood sugar (say, after a meal) or a drop (say, after an insulin

injection), then the glucose concentration reaches levels that are not contained in the range predicted by steady-state glucose concentrations. Some other approaches focus their attention at or near the equilibrium levels only. Hence, we developed a new approach that computes the actual bounds within which the glucose concentration varies. The spike response of the system and other dynamics can help biologists elucidate potentially harmful effects of certain diabetes treatment options. Such modeling can be used to evaluate different insulin regimens for models of Type 1 diabetic patients, for example, given models of tolerance of other cell types (such as those involved in vision). Some of the details of this work have been presented in [27, 28].

One of the central and surprisingly general properties that we inferred using the analysis of these various biological models is that of box invariance. A system is said to be box invariant if there exists a range for each state variable defining a box such that once the system enters that box, it remains inside it. We observed that continuous dynamical system models of several phenomena proposed in the literature have the box invariance property. For example, the models of the Delta-Notch signaling, B. Subtilis sporulation initiation, blood glucose metabolism, Lac Operon, and cell signaling exhibit this property. We studied the computational and control aspects of box invariance, and their application to problems of biological interest [28].

Simulation techniques complement the formal symbolic analysis approaches described above. We developed some new approaches for simulating stochastic models of biological systems. The main idea is that the two random choices in Gillespies' stochastic simulation algorithm (SSA) can be separated. Gillespie suggested that in stochastic simulations, there are two main sources of randomness: the choice of which reaction to fire next, and the time delay before that reaction completes. We found that the first, using only the choice of which reaction to fire next could be used alone (ignoring the random time delay in its firing), and we are careful regarding those random choices, we can obtain time-abstract stochastic simulations. These simulations can be obtained very efficiently and they yield useful qualitative information about the biological system in question [29].

Biochemical networks can contain thousands of reactions. Hence, scalability of analysis techniques is important. SRI International developed a new scalable approach for analyzing discrete models of biochemical networks by translation to the Boolean MaxSAT problem. The novelty here is that we make reactions central to the notion of a steady-state behavior. A steady-state behavior is defined by a subset of reactions that can be mutually consistently “on”, and yield no net motion of the key system properties. The attractiveness of our approach is that it is generic and applies to models coming from many different kinds of biological networks. Additionally, this approach is also flexible and allows encoding of knowledge specific to certain kinds of networks via suitable manipulation of the weights on the generic Boolean constraints. MaxSAT solvers find a maximum weight model of a set of weighted Boolean clauses. Using this approach, we have analyzed models of several of the phenomena described above. We have also applied it to metabolic pathways. Further details of this work can be found in [30].

4.1.4.2 *Bacillus anthracis*

Working together with other Bio-SPICE performers, including new teammates brought in from Vital Probes, a small business with expertise in proteomics of pathogenic organisms, SRI developed models of *Bacillus anthracis* metabolism, and partial models of the *Bacillus anthracis* secretome. We curated a fairly complete model of *Bacillus anthracis* metabolism by combining

several sources of data. First, as mentioned earlier, *Bacillus subtilis* is a human-safe close relative of *Bacillus anthracis*. Thus, we used preexisting models of *Bacillus subtilis* and other organisms curated in the literature and in Bio-SPICE BioWarehouse. We also used the pre-existing Pathologic tool developed by SRI International researchers. This tool takes a sequenced and annotated genome for an organism, and matches predicted and annotated genes found in the genome to known pathways in the metacyc database. The metacyc database consists of all pathways curated in the BioCyC system found in any organism. When several components of a metacyc-curated pathway are found to be present in the genome of an organism, we can then predict that the remaining features of that pathway are likely present as well. So we then perform a detailed and potentially noisy search through the genome of the organism in question attempting to fill the gaps in the discovered pathways. As a result of this search, we populate the pathway genome database for the new organism to the best of our ability. SRI International produced a high-quality detailed metabolic map for *Bacillus anthracis* in this way. This pathway genome database is now available to the Bio-SPICE community through the BioWarehouse. SRI International makes this pathway genome database available to all DoD researchers through the *Bacillus anthracis* link on BioCyc.org to anthracyc.

In collaboration with Vital Probes, SRI International developed time-series pathway animations. Given a pathway genome database such as anthracyc, one has a relatively complete static map of metabolism pathways. What is sometimes of interest to biologists such as those at Vital Probes and various DoD laboratories includes the dynamics of pathways. In particular, Vital Probes performed a series of experiments involving the interaction of *Bacillus anthracis* with human lung tissue (or a simulant thereof). Vital Probes performed a set of experiments, and at given timepoints, performed proteomic assays including 2D gels. 2D gels separate proteins by size and charge, resulting a more refined view of what proteins are present in a given sample than any one-dimensional view. In addition, for selected proteins, Vital Probes picked spots out of the 2D gels, and performed mass spectrometry.

4.1.5 The Project Web Community

One of the major challenges facing any project shared among several contributing organizations is how to manage information and maximize collaboration. To this end, SRI set up a web site at biospice.org [1] to support the development effort, track issues, share information, and distribute files.

Teleconferences were held on a regular basis to keep contributing parties on schedule with development goals. As the project matured, software demonstrations were conducted by using Virtual Network Computing (VNC) technology to share the presenter's computer desktop with remote project participants.

4.1.5.1 Web Site

4.1.5.1.1 Overview

Throughout the lifecycle of the Bio-SPICE project, a publicly accessible web site has supported the development, distribution and marketing of the dashboard and its contributed modules. The web site provides a common space for communication, file sharing, and collaboration between the universities and research organizations that contributed to the collection of tools that make up Bio-SPICE.

4.1.5.1.2 Site Framework and Web Server

Based on the PostNuke [31] content management system, the website handled website registration, user account creation, session management (log-ins/outs), and includes a flexible permissions system to create unique access privileges for separate user groups. By requiring a log-in and using SSL encryption for the website, we were able to secure project data while still providing easy access to information for project members and other interested, registered users. We used an Apache [32] web server hosted on a machine running Red Hat Linux [33] to host the Bio-SPICE web site at SRI.

4.1.5.1.3 Purpose

As the project matured, the primary purpose of the website transitioned with a changing user membership. Initially created as a developer's resource, the web site eventually provided downloads and manuals for new users of the Bio-SPICE software and eventually was completely revamped to provide a professional and informative public face for the final product. Each successive phase was designed meet the needs of a broader membership while maintaining usefulness to existing members and, most importantly, supporting the continued development of the project.

In earliest phase of the project, the web site was used primarily by the contributing software developers. This website was known as the "Community" website and the tools installed and maintained on the website were designed to support development community. Critical tools included the Announcements module, the Mantis issue, the Wiki pages, Static Documents, ViewCVS, and the Sandbox.

In Phase 2, new emphasis was placed on making the Bio-SPICE dashboard and contributed tools available as a download to non-developers. This required the creation of a completely separate web site – the Bio-SPICE "Users" web site. In this site the File Exchange and Static Docs modules were critical in order to provide registered users with file downloads and user guide documents. Also, a unique set of announcements was directed at this audience in order to isolate these members from the behind-the-scenes business of continued software development. The two-site configuration eventually proved to be a less than optimal solution since it required two separate accounts on each site for developers. The developers also had to remember two separate logins, although efforts were made to use the same login for both site. The web site management overhead also increased appreciably, since it required support personnel to maintain two accounts for each developer along with posting duplicate announcements and other information that pertained to the target audience of both sites.

In the final years of the project, the web site membership overwhelmingly consisted of software consumers and people that were simply curious about Bio-SPICE and what it does. It was time to start providing more information to casual web browsers who had either heard of the project from a colleague, or had simply stumbled across the site. Phase 3 saw the Community and Users web sites combined into one site with a new professional and appealing appearance, and a layout that provided a much needed improvement for visitors and members alike. This is the site you see today when you visit http://www.biospice.org. By making intelligent use of the PostNuke permissions system, unique web content could be presented for visitors, standard users, developers and administrators. Content affected by the new permissions scheme included the menu items, announcements, and file downloads. Lastly, a new interface for browsing

information on the Bio-SPICE tools was developed. It is discussed in more detail under Tools Data in the section below.

4.1.5.1.4 Tools

Announcements — This area of the website provided a simple means of highlighting important information. The announcements would appear in the center of the home page with a heading in bold and summary text below. The announcements would appear in reverse chronological order, placing the most recent at the top of the page.

Static Documents — The basic building block of a web site is the static html web page. Static documents are not controlled by the content management system but are simply regular web pages, sometimes legacy files from a preliminary web site, which are posted to the web while preserving the look and feel, and menu navigation capabilities created by the content management system. Examples of static documents on the Bio-SPICE website included the Dashboard Manual and the Developer's Manual, which were converted from printable documentation for posting to the web. These pages contained very few links, and most of those links were added-in after the conversion to html in order to improve access to referenced files.

Wiki — The wiki is a web page that can be edited on-the-fly through a web browser. The user simply clicks the 'edit' button on the page they wish to modify, changes the text in the edit frame, and clicks the 'save' button to implement the changes. These pages are useful for allowing users to post discussion comments and information to the website without requiring direct access to the web server. When preparing for a conference or meeting, we would post a top-level page containing the agenda or schedule for the event, with links to sub-pages for each item. The developers and contributors could then add in-depth information to their respective pages. Other uses include quick turnaround user guides that provided instructions for accessing the CVS repository. The wiki pages can also host downloads, place pictures in-line with text, and password protect an individual page or group of pages to keep it (them) private from the general website membership.

File Exchange — One of the key benefits of having a website is being able to distribute files for website members to download. This feature was fully exploited on the Bio-SPICE website to provide downloads for beta releases, stable and legacy releases of the Dashboard, as well as compatible Modules provided by the project contributors along with associated documentation. Downloadable versions of the Dashboard Manual and other documents were also available as well as installers for VNC, our web conferencing software. Due to the large size of the user base for Bio-SPICE, we chose to allow only administrators to upload new files or delete existing files.

View CVS — The document repository uses an open-source versioning system called CVS [9]. In order to add or modify files in the repository, a project developer would need an account on the CVS server. A checkout of the files (read-only access) was provided to all website members. Since most casual website users are unfamiliar with CVS and its command-line interface, we installed the ViewCVS [8] tool on the website to provide a web-based means of browsing the repository directory structure and downloading any individual file. If a user wanted a full copy of the repository, they would need to checkout the files using the command line or a GUI tool.

Bug Tracker — An essential requirement for any quality-controlled software development project is a means of submitting and tracking bugs. The Bio-SPICE dashboard and website both

benefited from the use of Mantis [36] to track and resolve issues related to each aspect of the project. Mantis, like most bug tracking tools, provides a means of a bug through a standard workflow from initial submission to closure. Along the way, a manager can assign a bug to a developer, set the priority or severity of the bug, set a deadline for resolution, or move it to a different category. When finished, a developer will set the status to 'resolved', after which an appropriate party will set the status to 'closed' after the resolution has been verified. Mantis also has a user-friendly home page the shows a list of bugs currently assigned to you, along with a color-coded status and severity of each bug, and a similar filterable list that initially shows all open bugs that the user has permission to view.

Sandbox — The Sandbox is essentially a file browser for the web server. It is a means of sharing files among web site members, similar in purpose to the File Exchange tool, only much less formal. Whereas the File Exchange was entirely managed by the website administrator, the Sandbox allowed any member to add or remove files or directories. It was a "use at your own risk" type of tool since there was no guarantee that someone would not delete your file, accidentally or otherwise. Luckily, that never happened and the Sandbox in its final form is full of files and directories that represent the open exchange of files throughout the whole project's lifespan.

Tools Data — In the final phase of the project a customized tool was developed to more clearly and simply display information for all of the contributed Modules that can be used with the Dashboard. Visitors to the website could browse a summary list and view a more detailed description by clicking a link. Website members could browse the same list, but were treated to additional links for downloading documentation and software. The tool includes a rather robust administrative interface for updating information and adding new tools.

Events Calendar — The events calendar would place events in a tabular formatted calendar, appearing much like a monthly flip calendar that you hang on the wall. Each event appeared in a color-coded box with a title and brief summary. The colors could be customized according to a category relating to they type of meeting or work group involved. Bio-SPICE had categories for Conferences, IFDEF, SEPDTF and Web Seminars, each with its own highlight color to more-easily identify them on the calendar.

FAQ — To address the frequently asked questions of the typical developer or software user, an FAQ section was set up on the web site. The FAQ included answers to questions for visitors asking how to join the Bio-SPICE project, developers inquiring about the mail lists and how to subscribe, how to attend the web seminars, and how to download the source code from the CVS repository using anonymous access, and quite a bit more.

4.1.5.1.5 Web Conferencing

A significant part of project collaboration involved not reading about another contributor's software tool, but actually seeing it in action. Regular meetings in person to demonstrate software would have been beneficial, but were cost prohibitive. As such, a web conferencing technology called VNC was adopted to help facilitate remote software demonstrations between the geographically distributed organizations.

VNC is a two-part software bundle that includes a server and a viewer. The person giving the presentation runs the server while remote attendees to the presentation use the viewer to see the

presenter's computer screen. When configured properly, the viewer can actually take control of the remote computer, although for this purpose the 'view only' option was used. This system proved to be an invaluable asset in a live demonstration that allowed attendees to ask questions in real time, directly to the developer of the tool, and to provide further development guidance to improve usability and fix bugs.

VNC was chosen over other desktop remoting options because it is cross-platform compatible and free. Platform independence allows a Macintosh, Windows, and Linux computer to participate in the web conference, regardless of the operating system the presenter is running. VNC is an open-source technology with many differing versions currently available [37].

4.1.5.1.6 Mail List Server

Mail lists are a quick and simple way to distribute email to all persons interested in a specific topic of discussion. For Bio-SPICE, we set up a separate mail list for each of the working groups, for general developer announcements, and for all web site members. Web site members could subscribe to a mail list by sending a request to the support group. Sending a message to all members of a mail list is as simple as typing "<list-name>@biospice.org" in the To: field while composing a message in your mail client. Mail lists created for the Bio-SPICE project included: Bio-SPICE-all, Bio-SPICE-dashboard-dev, Bio-SPICE-developers, Bio-SPICE-mdl and Bio-SPICE-sepdtf. We used MailMan [38], an open source, free and fully featured mail list server.

5 CONCLUSIONS, OBSERVATIONS AND LESSONS LEARNED

The program from its inception pursued a number of sometimes orthogonal goals with varying degrees of success. Overall, the program sought to create a community, extend research in a number of areas touching on systems biology, and to create an integrated suite of tools that would make cutting edge systems biology modeling and analytic tools accessible to the every day biology researcher.

5.1 PROGRAM CHALLENGES

While the program was very successful overall, two main areas remain to be addressed in the future. The successes of the program (highlighted in the next section) were achieved only after addressing many very serious potential impediments to program success. Key among these is the cultural gulf and language barriers between the several disciplines required to execute on the overall program vision. Facilitating cross-discipline discussions, education, and collaboration enabled a community to form around this program and to result in the successes that were achieved. Still, other challenges remained.

5.1.1 The Electronic Lab Notebook

The electronic notebook was envisioned as a way to publish and enable comment on repeatable experimental results in an electronic form. Its lack did not impact the basic integrating framework. A researcher could still use appropriate components from the suite of tools to build a model of interactions, test the model in a simulation engine and pass the results of simulation to a series of analytic tools. What the electronic notebook would have added would have been an overarching and familiar user interface that would have allowed the user to deal with experiments that are part of projects, maintaining the structure of the experiments, the dashboard

components and workflow used, links to publications, external data sources, and the researcher's own notes. Ultimately it was envisioned that the notebook would provide a way of unalterably and unambiguously tagging all of the items making up its content to support IP claims and similar concerns.

The electronic lab notebook provided the least research return if viewed from the perspective of extending the science under pinning the systems biology tools that the program was interested in. But its lack also meant that widespread adoption of the integrated tool suite was made that less likely by the traditional biologists who stood to benefit the most from the application of these tools. Depending upon which metric you applied, the failure to deliver an electronic lab notebook was either a very small shortcoming, or a barrier to adoption of the tool suite by a larger, non-systems biology research community.

5.1.2 Experimental Data Exchange Formats

Some of the most significant integration work being done was the development of exchange formats for modeling data and methods, and the development of exchange formats for the exchange of experimental data. The former efforts were very successful and drove the useful extension of the SBML, an open source, community supported standard for exchanging modeling data and methods. The efforts to develop useful exchange formats for experimental data exchange were less successful. For the initial period of the program, the only agreed to exchange format was a very open, context defined format involving a simple tuple structure. This was known as the time-series format. It required that component tool makers provide other mechanisms to appropriately map data they produced in time-series format so that it could be used by consuming components.

Toward the end of the program, some efforts were made to incorporate a version of the MIAME microarray data exchange format as another supported format. This effort was successful to a degree, though there still remained the larger issue that common experimental data exchange formats were not completely semantically annotated. Challenges remain in defining, adopting, and refining data exchange formats to enable more tools and more researchers to interoperate within the common integration framework. We continue to influence the community and point toward needs for additional standards and best practices in using them.

5.2 PROGRAM SUCCESSES

The overall successes of the Bio-SPICE Program included many exciting biological research results, new systems biology tools, an integration infrastructure, several integrated use cases demonstrating revolutionary uses of combinations of systems biology toolsets, and the creation of a community. It is a testament to the vision of the originators of the program that in the space of a few short years, the community bridged the daunting gap between hard-core biologists, physicists, computer scientists, mathematicians, and biological problem stakeholders. Remarkably, not only were we able to forge a community able to effectively communicate, we were able to work together toward common goals and produced a series of successful demonstrations of the program approach**.**

5.3 LESSONS LEARNED

- From the point of view of the integrator, it is much easier to manage the creation of an integrated software product if the integrator has some direct control over the component developers, either through a sub contract mechanism or other equivalent mechanism
- Flexibility can be very useful in seizing opportunities presented in research, but a successful multi-year software development effort needs the stability of an agreed to set of requirements and managerial change control mechanism that balances cost against benefit of proposed changes.
- The benefits of using research to help nurture a community of interest are difficult to quantify but they are very real
- If one of the outcomes of a program is a robust and usable software product, the program management must not underestimate the costs of creating that product from research software. The quality of software received as contributions on this project varied widely from thoroughly professional to idiosyncratic hacks.
- The difficulty of bridging the gaps between several disciplines of research should not be underestimated.
- The importance of forming a thriving community and team spirit should also not be underestimated.
- Driving systems biology research from the perspective of real-world biological problems of interest to DoD was a rousing success. New interactions between research teams driven by requirements of considering those real-world problems led to new insights, new tools, and new uses of existing tools in combination to help address the problems of interest. This was a very positive force in the program.
- The leadership from the government on this topic was and is crucial to continuing successes in systems biology as needed to address urgent DoD problems.
- A major upheaval in how biological research is carried out is underway. Almost simultaneously, the availability of (limited kinds of) high-throughput timecourse data, large databases (particularly of genomic and proteomic information), and capable computational tools are enabling systems biology researchers to address problems at unprecedented scales of complexity. Although enabled by these unfolding revolutions, biological researchers face a daunting challenge to harness them, interact with large multidisciplinary teams, and a myriad of heterogeneous tools. The Bio-SPICE program has helped to address this challenge through the creation of an integration framework, the demonstration of these tools in use on problems of real urgency and importance to the DoD, and the creation of community that can grow and influence the future of biological research through outreach, publication, and proselytization.

6 REFERENCES

[1] Bio-SPICE: http://www.biospice.org . URL working as of June 20, 2007. See also http://sourceforge.net/projects/biospice.

[2] Koster, K., Gilman, A., and Arkin, A. (2003). NetBeans platform and custom code modules. Submitted to the sri-arkin BioSPICE Dashboard code repository, with subsequent revisions by the Arkin Laboratory, University of California, Berkeley.

[3] OAA Homepage: http://www.ai.sri.com/~oaa/. URL working as of June 19, 2007.

[4] SBML Homepage: http://sbml.org/index.psp. URL working as of June 19, 2007.

[5] ENZYME Homepage: http://ca.expasy.org/enzyme/. URL Working as of June 19, 2007.

[6] KEGG Homepage: http://www.genome.jp/kegg/. URL Working as of June 19, 2007.

[7] BioCyc Homepage: http://www.biocyc.org/. URL Working as of June 19, 2007.

[8] UniProt Homepage: http://www.pir.uniprot.org/. URL working as of June 19, 2007.

[9] GenBank Homepage: http://www.ncbi.nlm.nih.gov/Genbank/index.html. URL working as of June 19, 2007.

[10] NCBI Taxonomy Homepage: http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/. URL working as of June 19, 2007.

[11] CMR Homepage: http://cmr.tigr.org/tigr-scripts/CMR/CmrHomePage.cgi. URL working as of June 19, 2007.

[12] Gene Ontology Homepage: http://www.geneontology.org/. URL working as of June 19, 2007.

[13] MATLAB Homepage: http://www.mathworks.com/. URL working as of June 19, 2007.

[14] ptPlot Homepage: http://ptolemy.berkeley.edu/java/ptplot/. URL working as of June 19, 2007.

[15] SBW Homepage: http://sbw.sourceforge.net/. URL working as of June 22, 2007.

[16] MIME information can be found at: http://www.mhonarc.org/~ehood/MIME/MIME.html. Working as of June 25, 2007.

[17] Conventions for Media Types for MIME-type naming are defined in can be found at: http://www.mhonarc.org/~ehood/MIME/2046/rfc2046.html. URL working as of June 25, 2007.

[18] Conventions for Registration Procedures for MIME-type naming can be found at: http://www.mhonarc.org/~ehood/MIME/2048/rfc2048.html. URL working as of June 25, 2007.

[19] NetBeans directory structure specification: http://autoupdate.NetBeans.org/nbm/nbm_package.html. URL working as of June 26, 2007.

[20] Additional information on NetBeans manifest format and the module JAR file: http://www.NetBeans.org/. URL working as of June 26, 2007.

[21] More information on NetBeans lookup system can be found at: http://openide.NetBeans.org/lookup/index.html. URL working as of June 26, 2007.

[22] Definitions of XSD primitive types can be found at: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#built-in-datatypes. URL working as of June 26, 2007.

[23] Lee, T. J., Pouliot, Y., Wagner, V., Gupta, P., Stringer-Calvert, D. W. J., Tenenbaum, J. D., and Karp, P. D. 2006. BioWarehouse; a bioinformatic database warehouse toolkit. *BMC BioInformatics2006*, 7:170. Article is available from: http://www.biomedcentral.com/1471-2105/7/170 . URL working as of June 26, 2007.

[24] URL for Professor John Tyson's Computational Cell Biology Lab: http://mpf.biol.vt.edu/Tyson%20Lab.html. URL working as of June 26, 2007.

[25] URL for Professor Frank Doyle's Doyle Group: http://doyle.chemengr.ucsb.edu/. URL working as of June 26, 2007.

[26] Ghosh, R., Tiwari, A, and Tomlin, C. 2003. Automated Symbolic Reachability Analysis with Application to Delta-Notch Signaling Automata. *Hybrid Systems: Computation and Control.* Ed. Maler, O. and Pnueli, Philadelphia:Springer, April. 233-248.

[27] Lincoln, P. and Tiwari, A. 2004. Symbolic systems biology: Hybrid modeling and analysis of biological networks. *Hybrid Systems: Computation and Control, 7th International Workshop.*, Philadelphia:Springer, March 25-27. 660-672.

[28] Abate,A. and Tiwari,A. 2006. Box invariance of hybrid and switched systems. *2nd IFAC Conf. on Analysis and Design of Hybrid Systems.* Alghero, Italy, June 7-9. 359-364.

[29] Abate, A., Bai, Y., Talcott, C., and Tiwari, A. *Quantitative and Probabilistic Modeling in Pathway Logic*. Submitted for publication 2007.

[30] Tiwari, A., Talcott, C., Knapp, M., Lincoln, P. and Lauderoute, K. *Analyzing Biochemical Pathways using SAT-based Approaches*. Submitted for publication 2007.

[31] PostNuke: http://www.postnuke.com. URL working as of June 26, 2007.

[32] Apache: http://httpd.apache.org/. URL working as of June 26, 2007.

[33] Red Hat Linux: http://www.redhat.com/. URL working as of June 26, 2007.

[34] CVS: http://www.nongnu.org/cvs/. URL working as of June 26, 2007.

[35] ViewCVS: http://www.viewvc.org/. URL working as of June 26, 2007.

[36] Mantis: http://www.mantisbt.org/. URL working as of June 26, 2007.

[37] VNC Sources: http://www.uvnc.com/; http://sourceforge.net/projects/vnc-tight/; http://www.realvnc.com/. URLs working as of June 26, 2007.

[38] MailMan: http://www.gnu.org/software/mailman/. URL working as of June 26, 2007.

APPENDIX A—DETAILS OF KEY ACCOMPLISHMENTS/ PROGRAM HISTORY

This program was essentially an integration task. Its principal technical objective was to develop and evolve the integration environment, and as other program performers developed tools, to validate their ability to work within the integration framework. There were seven major releases of the integration environment and additional minor releases. The major releases were generally timed to coincide with the semiannual PI conferences. The table below provides a capsule history of the core Bio-SPICE releases, as well as well as key accomplishments within the integration, Biowarehouse, tool creation and community creation efforts.

Table 4. Bio-SPICE Development Timeline.

Date	Integration	Biowarehouse	Dashboard Tools	Community Services
31Dec 2001	OAA adopted as remote service facilitator			http://biospice.org established DARPA Open Source License posted
31Mar 2002	Began examining candidate exchange formats. Tasked to provide significant support for May 2002 PI meeting	Took on development of community supporting bio informatics db based upon community specified databases		All PI's registered on web site
31Jul 2002	Bio-SPICE 1.0 -22May Bio-SPICE 1.01 – 31 May Bio-SPICE 1.1 – 14 June 15 contributed packages released OAA updated for Bio-SPICE with direct-connect functionality	Work continued first release of data warehouse		125 registered users on web site Site provides access to software information, collaboration and file exchange functionality, and access to SRI provided software development services including code repository viewing, issue tracker access, etc.
11Jan 2003	Bio-SPICE 2.0 – released at Dec 2002 PI Conference Demo of OAA wrapped components at Dec PI Conference All submitted Bio-SPICE agents validated Release 2.0 included installers for Windows and Linux Conceptual Dashboard interface demonstrated	SwissProt/ Enzyme and BioCyc loaders for Biowarehouse v 1.0 released	OAA-to-SBW bridge agent released	Added quick index of all software Established Mailman based mailing lists with web interface Established separate information pages and collaboration spaces for the MDL and EWG working groups Presentations from previous conferences available on site 201 registered users
12Apr 2003	In conjunction with LBL began development of			Developers and information web sites merged. Data

Date	Integration	Biowarehouse	Dashboard Tools	Community Services
	Dashboard based upon NetBeans framework Minimum contribution standards est. Contributions to May release must be OAA wrapped and consume or produce SBML wrapped data.			partitioned by permissions system Bio-SPICE-pi, Bio-SPICE-developers, Bio-SPICE-all and several special purpose mailing lists instituted to support program 329 registered users.
29Jun 2003	Bio-SPICE 3.0 released at May 03 PI conference First production version of NetBeans Dashboard platform released at May 03 PI conference Platform demonstrated using tools from four different teams Model Definition Language based upon SBML2 accepted. Dashboard analyzer API published		Dashboard OAA bridge released	400 + registered users
27Sep 2003	Created Dashboard demonstration use case to focus continuing development efforts, including determining which tools would be used in the use case Released installer based version of Dashboard that wraps both Dashboard and OAA facilitator. Enabled the remote software update capability of the Dashboard and established an update server for Dashboard core and modules	Released a Biowarehouse data query agent integrated directly into the Dashboard. Developed schema change to support MIAME format micro-array data Established publicly available Biowarehouse server. Began work to port Biowarehouse loaders to support MySQL as well as Oracle	Released Dashboard-MATLAB bridge to beta testing	488 registered users 55,000+ page hits on web site since site inception 400+ downloads of Bio-SPICE release 3.0 Published Bio-SPICE article in OMICS, Journal of Integrative Biology.
31Dec 2003	Winter use case near completion, involves use of 8 tools from 5 contributors within the Dashboard framework	Biowarehouse 2.0 released. Winter use case supported by the public Biowarehouse All Biowarehouse loaders successfully ported to MySQL	Dashboard MATLAB bridge released as the BioMAT bridge	553 registered users
27Mar 2004	Bio-SPICE 4.0 based upon the Dashboard released at	Work begun to add Genbank loader to		636 registered users

Date	Integration	Biowarehouse	Dashboard Tools	Community Services
	Feb 04 PI conference Winter use case successfully demonstrated at Feb 04 PI conference Use case demonstrated at DARPATech, Mar04	Biowarehouse		
29Jun 2004	Upgraded OAA for better support of peer to peer connections among distributed components Further work done to adopt and extend SBML2 as the Bio-SPICE model definition language EWG established time-series and MIAME micro-array formats as Dashboard standards SEPDTF published a Dashboard API for use by tool contributors Second round of use case focused development underway	Biowarehouse 2.1 released	Released module to access ptPlot from Dashboard Released bi-directional SBW bridge	770 registered users On-line documentation for Dashboard and BioMAT bridge added to web site Began program of user focused web seminars
29Sep 2004		Next release of Biowarehouse scheduled for Dec 04	Fully configurable Dashboard analyzer module (XML wrapper) released BioMAT bridge gui released	850 registered users Continued web seminars addressing basic Dashboard use and orientation, use case display, and tool demonstration
25Dec 2004	Bio-SPICE 5.0 released 7 Oct, prior to the Oct PI conference SRI hosted three day Dashboard architectural review All program performers participated in one of six demonstration use cases	Biowarehouse 3.0 released	New and updated versions of contributor tools release with Bio-SPICE 5.0	950 registered users Web page hits greater than 140,000 Work begun with SAInc to upgrade look and feel of current website
15Apr 2005	Bio-SPICE 6.0 released early to stabilize the core for tool contributors developing for the May 05 PI conference NetBeans framework upgraded			1030 registered users

Date	Integration	Biowarehouse	Dashboard Tools	Community Services
	CVS access granted to key contributors outside of SRI and LBL			
30Jun 2005	Three patches released for Bio-SPICE 6.0. Effort initiated to resolve outstanding issue tracker issues. 16 resolved, 86 outstanding Revision and extension of Dashboard documentation Improvement of workflow engine Improvement to UI Beta port of Dashboard to run under Mac OSX	Biowarehouse 3.5 released		1173 registered users Upgraded web site on line 5/04 Extensive support to performer developers in wrapping or making tools accessible to Bio-SPICE
30Sep 2005	Mac OSX port of Dashboard released Core Dashboard enhancements including adding the ability to cut, copy and paste workflow objects, and moving to support Java 1.5.			1200+ registered users Bio-SPICE project established on Sourceforge pursuant to transition strategy to release Dashboard into the public domain
31Dec 2005	Bio-SPICE 7.0 released at Nov 05 PI conference Six use cases demonstrated at Nov 05 PI conference			1240 registered users Bio-SPICE repositories moved to Sourceforge servers and accessible through the public Bio-SPICE project hosted on Sourceforge. Write access to these repositories limited to current development team and selected others
31Mar 2006	SRI provides support to current developers and open source users SRI developers continue to address issues in issue tracker. SRI developers continue to review, revise and update project documentation			Project web site content reviewed with COTR to identify material which is publicly releasable before opening the web site
30Jun 2006	SRI provides support to current developers and open source users SRI developers continue to address issues in issue	Development of additional loaders continues Biowarehouse 3.6	SRI continues to integrate pathway analysis tools into	1349 registered users

Date	Integration	Biowarehouse	Dashboard Tools	Community Services
	tracker. SRI developers continue to review, revise and update project documentation	released 4/26	Bio-SPICE Dashboard. Analysis of datasets of samples from WRIAR continues	
30Sep 2006	SRI provides support to current developers and open source users SRI developers continue to address issues in issue tracker. SRI developers continue to review, revise and update project documentation	Development of additional loaders continues Biowarehouse 3.7 released 8/31/06 Biowarehouse 3.8 released 9/8/06		
30Nov 2006	SRI provides support to current developers and open source users SRI developers continue to address issues in issue tracker. SRI developers continue to review, revise and update project documentation	Biowarehouse 3.9 released 10/5/06 Biowarehouse 4.0 released 11/30/06		1431 registered users 240,000 page hits Project mailing lists except support@biospice.org discontinued. All remaining issues including resolved issues moved to Sourceforge issue tracker. biospice.org name rights purchased for 10 years biospice.org web site to be hosted by SRI after project completion

APPENDIX B—CONTRIBUTED TOOLS

The following tools were contributed to the Bio-SPICE program under a series of different contract vehicles. Individuals interested in the current state of a particular tool should contact the responsible individual shown in the tool detail sheet.

Table 5. Bio-SPICE Program Contributed Tools.

MODULE	INSTITUTION
2DGrapher	SRI
Add Column to Time series	UTK
Biodata Viewer	SRI
BioGrid	UTK
BioMat Bridge	SRI
BioNetS	UNC
BioPack	VT
BioSens	UCSB
BioSketchPad	U Penn/BBN
BioSmokey	UTK
BioSpreadsheet	UTK
BioWarehouse	SRI
BioWarehouse Query	SRI
BioWarehouse2SBML	Harvard
BioWave	NYU
CellX	Indiana
Charon	U Penn
Clone Updater	TJU
CoBi	CFDRC
Convert Data To Graph	UTK
DBAgent	SRI
DNAgen	SUNY Geneseo
ESS	UTK
Fluxor Computational Analyzer	Harvard
Fluxor Spreadsheet	Harvard
FTF	
GCMConverter	
GCMMerger	
GeneCite	WRAIR
GeneScreen	UCLA
GeneWays	Columbia
Get Column from Time Series	UTK
Get Row from TimeSeries	UTK
Graph Viewer	UTK
Graphviz	LBL
Homolog Finder	LBL

Hybrid Automata Symbolic Reachability Tool	Stanford
IcDNA	UCLA
Jdesigner	KGI
JigCell Tools	VT
JournalMine ActionQuery	Columbia
Karyote Cell Analyzer (KCA)	Indiana
Karyote Genome Analyzer (KAGAN)	Indiana
Merge Timeseries Columns	UTK
Merge Timeseries Rows	UTK
MetaCluster	TJU
MetaTool	KGI
MIAMESpice	UCLA
Monod	MSI
Network Component Analysis (NCA)	UCLA
NYUMAD	NYU
NYUSIM	NYU
Octave Bridge	UTK
Oscill8 Tools	
PAINT	TJU
PathwayBuilder	LBL
PathwayScreen	WRAIR
PlotML Translator	SRI
PtPlot	SRI
QIS	QIS
Reactome	
Render Matlab Simulation	LBL
Render SBML	LBL
SAL	
Sample Timeseries	UTK
SBML TO Graphviz Dot	LBL
SBML to PathwayBuilder	LBL
SBML to laid-out Graphviz DOT	LBL

SBML:SCC to PathwayBuilder	LBL
SBWMatlab	KGI
Strongly Connected Component (SCC) Finder	LBL
Sensitivity Analyzer	LBL
Simpathica	NYU
SOS Tools	Cal Tech
Symbolic Analysis Toolbox	Stanford
Tab Dlimited Text Converter	SRI
TableView	SRI
Timeseries to Text	UTK
TimeSeries To ZipFile Converter	SRI
Vibrio	U Penn

MODULE NAME

2DGrapher

INSTITUTION

SRI

MODULE NAME

Add Column to Timeseries

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

This module adds an additional column of data to a Timeseries. The module takes a Timeseries as input and a produces a Timeseries as output. Double-clicking the module allows configuration of two additional parameters, “New Column Header” and “New Column Data”, which specify the header and data for the new column.

MODULE NAME

Biodata Viewer

INSTITUTION

SRI International

MODULE NAME

BioGrid

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UTK/biogrid.pdf

http://biocomp.ece.utk.edu/tools.html

cvs-repository\tools\utenn\BioGrid\BioGrid3_usecase_7-04

SUMMARY

BioGrid is a set of tools used to distribute simulation and analysis jobs across a

cluster of computers. The basic use model for BioGrid is:

1. Create an SBML model in BioSpreadsheet or another SBML model creation tool (i.e. JigCell, BioSketchpad, etc.).
2. Import the model into the BioGrid Client.
3. Configure a set of parameters that you wish to sweep.
4. Configure a set of post-processing commands.

BioGrid then creates individual SBML files for each model parameter, simulates each model using exact stochastic simulation, and post-processes the output data using octave. When all jobs are complete, BioGrid creates a web page that contains the results of your simulation run and links to all of the SBML models and data files created by the run. Because BioGrid runs the jobs in parallel, performance is greatly improved over running the simulations on a local machine. BioGrid can be especially useful for:

- Users trying to configure model parameters to get their simulation results to match a set of observed data.
- Users who wish to analyze the sensitivity of their model parameters.
- Users who need to run several simulation runs with multiple random seed values.

BioGrid's exact stochastic simulator employs an algorithm similar to Gillespie's Stochastic Simulation Algorithm. For more information on this technique, users are encouraged to consult the following references:

Gillespie, D. T. 1977. Exact Stochastic Simulation of Coupled Chemical Reactions. Journal of Physical Chemistry, vol. 81, pp. 2340-2361.

Gibson, M. A., and J. Bruck. 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. Journal of Physical Chemistry A, vol. 104, pp. 1876-1889.

MODULE NAME

BioMat Bridge

INSTITUTION

SRI International

WEB SITE

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/SRI/MATLABAnalyzer/index.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/SRI/MATLABAnalyzer/BioMat_Bridge_PI_Oct_2004.pdf

cvs-repository\tools\sri\BioMatBridge

SUMMARY

The BioMat Bridge is a polymorphic workflow object that bridges between the Bio-SPICE dashboard and Matlab™.

"Motivation"

Matlab is a widely used system for scientific programming and as such enjoys a wide audience among biologists and bioinformaticians. In order to make the Bio-SPICE platform accessible to a user community that is already exposed to Matlab and to leverage existing and legacy code, it is imperative to integrate Matlab with the Bio-SPICE dashboard

"Concepts"

The BioMat Bridge enables a workflow in the Bio-SPICE dashboard to enter (and exit) Matlab's computation engine. It allows Matlab functions to seamlessly interact with workflow objects in the dashboard without any restrictions on what the Matlab code actually does. Depending on how it is configured, a BioMat Bridge object may perform analysis, visualization, simulations, collect user input, generate data, etc. Anything that can be done with Matlab can be integrated into a workflow using the BioMat Bridge. As such, the BioMat Bridge provides more generic functionality than other workflow objects.

We have worked hard to engineer an interface to Matlab that is flexible, powerful and easy to use. However, supporting this generic functionality within the Bio-SPICE dashboard also leads to some unique behavior compared to other objects.

In order to better understand how the BioMat Bridge works and how to use it effectively, we would like to introduce a few important concepts.

MODULE NAME

BioNetS

INSTITUTION

University of North Carolina

WEB SITE

http://x.amath.unc.edu/BioNetS

cvs-repository\tools\unc\BioNetS\BioNetS_usecase_7-04\BioNetS_Submission\Windows\README.TXT

SUMMARY

"Biochemical Network Stochastic Simulator"

An easy to use and blazingly fast stochastic ODE solver, using Gillespie, chemical Langevin, and any hybrid of the two.

We have developed the software package Biochemical Network Stochastic Simulator (BioNetS) for efficiently and accurately simulating stochastic models of biochemical networks. BioNetS has a graphical user interface that allows models to be entered in a straightforward manner, and allows the user to specify the type of random variable (discrete or continuous) for each chemical species in the network. The discrete variables are simulated using an optimized implementation of the Gillespie algorithm. For the continuous random variables, BioNetS constructs and numerically solves the appropriate chemical Langevin equations. The software package has been designed to scale efficiently with network size, thereby allowing large systems to be studied.

BioNetS is a code generator. As input, you enter in the reactions using a standard spreadsheet like interface, or import them from an SBML file. Then you set which type of solver you want and which optional statistics should be accumulated during the run. You can then run the simulation from within BioNetS, export the executable for use from Matlab or DataTank or the unix shell. You can even view the C++ source code that gets generated and incorporate it into your own program.

Because the code is generated for each reaction system, and with the exact reactions statistics you want to compute, it is possible to tailor the code to the specific system. This allows code optimizations that a general purpose solver would not be able to do, and runs much faster. On a recent computer the Gillespie solver takes around five million steps per second and can use every Gillespie step to accumulate histograms.

BioNetS on the Mac generates graphics that can be copied as pdf documents and pasted into presentations and publications.

We want BioNetS to be a useful tool for working biologists as well as modelers. If you encounter problems, if you have questions, or if there are capabilities you wish the software had, please contact us.

MODULE NAME

BioPack

INSTITUTION

Virginia Polytechnic Institute and State University

WEB SITE

http://mpf.biol.vt.edu/~jzwolak/biopack/

http://jigcell.biol.vt.edu/parameter.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/VaTech/biopack.pdf

cvs-repository\tools\vatech\biopack\biopack_9-04\documentation.pdf

SUMMARY

"BioPack Parameter Estimation Demonstration"

JigCell needs a simulator and a parameter estimator. BioPack is the Jig Cell parameter estimation tool.

JigCell may want to call just the simulator or just the parameter estimator. Means must be present to do so. JigCell may want to call the simulator multiple times with different parameters. The code doesn't need to support easy calls for multiple simulations. JigCell currently cannot take advantage of such functionality and already has implemented multiple simulations of the same model through the Run Manager. Functionality does exist in Biopack to call just the simulator or parameter estimator.

This document provides instructions to install the Jig Cell BioPack parameter estimator software, along with a walkthrough demonstrating its basic use. In the future, the parameter estimator will be fully integrated with Jig Cell.

At this time, BioPack is only supported under Linux.

September, 2004

MODULE NAME

BioSens

INSTITUTION

University of California, Santa Barbara

WEB SITE

http://www.chemengr.ucsb.edu/~ceweb/faculty/doyle/biosens/BioSens_User_Guide.htm

http://www.engineering.ucsb.edu/~ceweb/faculty/doyle/docs/soft.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UCSB/BioSens.pdf

cvs-repository\tools\ucsb\BioSenS\BioSenS_9-04\BioSens_User_Guide.pdf

SUMMARY

"BioSens Sensitivity Analysis Toolkit"

BioSens provides a sensitivity analysis toolkit for Bio-SPICE through the BioMat Bridge. Sensitivity analysis investigates the changes in the system outputs or performance with respect to the parameter variations, which are quantified by the sensitivity coefficients S. Mathematically, the sensitivity coefficients are the first order derivatives of the outputs with respect to the parameters that affect the system dynamics. The toolkit implements centered difference approximation to compute the sensitivity coefficients.

The parameter perturbation magnitude Δp should be small enough to ensure small truncation error in the finite difference approximation, and large enough to reduce dependence on the simulation inaccuracies.

BioSens also includes Fisher Information Matrix (FIM)-based sensitivity analysis. Though the FIM was originally used to represent the amount of information contained in a given set of signals/measurements about the model parameters, it also can be interpreted as a consolidation of the system sensitivities. In this case, the FIM takes into account the underlying distributive nature of states or measurements due to the inherent stochasticity of cellular processes involving reactions with low copy number of substrates (McAdams, H. H. and A. Arkin, Trends Genet. 15:65-9, 1999) as well as noise in the measurements. The FIM-based sensitivity measures can give insights into the robustness and fragility trade off in biological regulatory structures based on the rank-ordering of the sensitivities (Stelling, J., E. D. Gilles and F. J. Doyle III, PNAS.101:13210-15, 2004).

For ease of use, BioSens functionalities are accessible through a Matlab graphical user interface.

MODULE NAME

BioSketchPad

INSTITUTIONS

University of Pennsylvania

BBN Technologies

WEB SITE

http://www.cis.upenn.edu/biocomp/new_html/biosketch.php3

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/BBN/biosketchpad.pdf

cvs-repository\tools\bbn\biosketchpad\BioSketchpad_web-downloads\docs

SUMMARY

The Bio Sketch Pad is a graphical editor for users to construct models of reaction pathways common in biological systems. When the sketch pad is appropriately configured, these models can be converted to forms supported by external simulation packages and executed in the simulation. Release 1.0 of the sketch pad supports conversion to the model format used by CHARON, a simulation tool developed at the University of Pennsylvania.

MODULE NAME

BioSmokey

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UTK/biosmokey.pdf

cvs-repository\tools\utenn\BioSmokey

SUMMARY

BioSmokey provides an easy well to manipulate time series data, with functions that include: column count, tab delimited conversion, column deletion, get column, get rows, row count, and sample.

MODULE NAME

BioSpreadsheet

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

cvs-repository\tools\utenn\BioSpreadsheet

SUMMARY

BioSpreadsheet is a biochemical model editor that can read and write SBML model files or BioSpreadsheet model files. The software can be used to generate a new SBML model by running the following workflow. The software can also read an SBML model or a BioSpreadsheet model at startup by connecting the model to the input of the module.

MODULE NAME

BioWarehouse

INSTITUTION

SRI International

WEB SITE

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/SRI/BioWarehouse/index.html

cvs-repository\tools\sri\warehouse\release-3.0\doc\index.html

SUMMARY

BioWarehouse is a component of the Bio-SPICE project. BioWarehouse is an open-source software environment for integrating a set of biological databases into a single physical database management system for data management, mining, and exploration.

Key features of BioWarehouse:

- A relational database schema that models important bioinformatics datatypes
- BioWarehouse instances can be implemented using either the Oracle or MySQL database management systems
- A collection of loader programs that populate the warehouse with data from public biological databases
- The loader programs transform the syntax of the source databases into relational form, and transform the diverse semantics of the source database into the common semantics of the BioWarehouse schema.

MODULE NAME

BioWarehouse Query

INSTITUTION

SRI International

WEB SITE

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/SRI/DW_Query_Analyzer/Warehouse_Query_Analyzer.html

cvs-repository\tools\sri\warehouse-query\doc\javahelp\com\sri\biospice\datawarehouse\BioWarehouse_Query_Analyzer.html

SUMMARY

The Warehouse Query Analyzer acts as a database proxy for the MySQL Bio-SPICE Biowarehouse. The Analyzer can accept any standard MySQL query as input and returns the result of the query in a TimeSeries format. The resulting TimeSeries can be passed onto another analyzer or viewed as a table using the Table Viewer.

Note: Because TimeSeries is a text-based format, it cannot include raw binary data. For this reason, all binary data returned by the Warehouse Query Analyzer is encoded in base-64 format.

MODULE NAME

Biowarehouse2SBML

INSTITUTION

Harvard University

WEB SITE

cvs-repository\tools\harvard\BioWarehouse\Biowarehouse2SBML_9-04\doc\Biowarehouse2SBML_Documentation.doc

SUMMARY

The Biowarehouse2SBML analyzer is provided for converting metabolic reaction datasets stored in the BioWarehouse from SRI into SBML files that can be used by the FluxorSpreadsheet analyzer to help prepare a list of reactions for a given dataset into an annotated SBML file that can be used as input to the Fluxor computational analyzer for carrying out flux analysis of a reaction system. The SBML produced by Biowarehouse2SBML can also be provided to the metatool analyzer from the Keck Institute to carry out elementary flux path analysis or any other tool that can operate on an SBML file as input.

August 2004

MODULE NAME

BioWave

INSTITUTION

New York University

WEB SITE

http://bioinformatics.nyu.edu/Projects/DARPA-BioComp/index.shtml

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/NYU/BioWave.txt

cvs-repository\tools\nyu\BioWave

SUMMARY

The present directory contains several Matlab (tm) 6.x files that make up a Graphical User Interface to a Time Series Time Frequency analysis tool capable of grouping different time-indexed data sets. The intent of this work is to provide automated function classifiers AFC, these AFCs shall sort a large number of functions based on the statistics of non-linear transformations. The intent is to develop a set of transformations which classify the function space according to qualitative features.

MODULE NAME

CellX

INSTITUTION

Indiana University

WEB SITE

http://biodynamics.indiana.edu/CellModeling/AboutCellX.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Indiana/CellX.pdf

cvs-repository\tools\uindiana\CellX

SUMMARY

CellX is a cell simulator that accounts for reaction and transport processes and the attendant intracellular gradients of composition. CellX is multi-dimensional − i.e. reaction-transport equations are solved along fibrils (1-D), on membranes (2-D) and within bulk media (3-D), all simultaneously and with full coupling that accounts for molecular exchange between these domains. CellX is a multi-scale simulator − overcoming omnipresent technical difficulties associated with widely separated timescales between the various biochemical reactions and with slow versus fast diffusing species. Equations are solved on finite element grids to yield the timecourse of cell state.

In the implementation provided here we include reactions and transport on the inner surface of the outer membrane of a bacterium; in particular the model accounts for surface processes associated with the dynamics of Min proteins and their transport within, and exchange with the cell's interior continuum. In more advanced Bio-SPICE releases of CellX the user will be able to introduce more general networks of reactions in the cellular interior, on or within membranes and along fibrils. For eukaryotic cells, complex intracellular structure will also be allowed. Those interested in these features can inquire ccvt@indiana.edu. Like a living cell with its multi-dimensional complexity, CellX is not easy to master.

Future releases will build in more user-friendly features (e.g., gridding, graphics, gui). For a CellX preview with easily controlled input and 3-D graphics/movies as output, go to biodynamics.indiana.edu. On the Cell Modeling page and you may try the CellX option. CellX has been developed at the Center for Cell and Virus Theory at Indiana University.

MODULE NAME

Charon

INSTITUTION

University of Pennsylvania

WEB SITE

http://www.cis.upenn.edu/mobies/charon/

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/UPenn/Charon_help/index.html

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/UPenn/charon_commands.txt

cvs-repository\tools\upenn\charon

SUMMARY

CHARON is a language for modular specification of interacting hybrid systems based on the notions of agent and mode. For hierarchical description of the system architecture, CHARON provides the operations of instantiation, hiding, and parallel composition on agents, which can be used to build a complex agent from other agents. The discrete and continuous behaviors of an agent are described using modes. For hierarchical description of the behavior of an agent, CHARON supports the operations of instantiation and nesting of modes. Furthermore, features such as weak preemption, history retention, and externally defined Java functions, facilitate the description of complex discrete behavior. Continuous behavior can be specified using differential as well as algebraic constraints, and invariants restricting the flow spaces, all of which can be declared at various levels of the hierarchy. The modular structure of the language is not merely syntactic, but also reflected in the semantics so that it can be exploited during analysis.

MODULE NAME

Clone Updater

INSTITUTION

Thomas Jefferson University

WEB SITE

http://www.dbi.tju.edu/tools/biospice/dbi_tju_feb_23/#cloneupdater

http://www.dbi.tju.edu/cloneupdater/html/template.php

http://www.dbi.tju.edu/tools/biospice/

http://www.dbi.tju.edu/tools/biospice/current_release/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/TJU/tju_tools.pdf

cvs-repository\tools\tju\cloneupdater

SUMMARY

The biological problem we study is unraveling the transcriptional regulatory networks underlying various cellular functions, specifically in mammalian systems. System-wide high throughput gene/protein expression data enables us to analyze gene regulation, in particular co-regulation, at a system level. We have developed various tools that enable biologists to integrate functional genomics data, for example from microarray-based gene expression analysis, with genomic sequence data to carry out transcriptional regulatory network analysis (TRNA).

Gene annotation is one of the most important data classes used in the postgenomic era. CloneUpdater provides the biologist an easy to use path to the most "up to date" annotation of genes and associated reagents such as EST clones. It is compatible with many different types of identifiers and provides a large number of options for updating preexisting annotations or adding new annotations to user-provided identifiers from many different databases including UniGene, LocusLink, RefSeq, and more. Processing more than 50 identifiers per second, CloneUpdater is fast enough to be used for one gene identifier or tens of thousands. In addition, CloneUpdater has the important ability to find identifiers that represent the same gene and thereby highlight redundancies in large reagent collections.

In the context of analyzing gene regulatory networks, CloneUpdater can be used as a preprocessor for PAINT: it can eliminate redundant clones, and update annotations that PAINT can process. This Dashboard release of CloneUpdater features a GUI that offers the same features as the online version. More information about CloneUpdater is available at the same URL.

MODULE NAME

CoBi

INSTITUTION

CFD Research Corporation

CONTACTS

WEB SITE

SUMMARY

Computational Biology, CoBi, tools is a software framework for 4D multiscale modeling of cell and tissue biology developed by CFDRC. CoBi solves partial differential equations for fluid flow, and multiple species transport including advection, diffusion, reaction on any type of grids (structured, unstructured even polyhedral elements). At present the mesh is read as an input. User input also specifies reaction kinetics pathways for various spatial zones (cytosol, membrane, ECM,..). In the next release we will provide tools for mesh generation and graphical model setup directly from microscopy images and interfaces to open source geometrical modeling tools. CoBi creates an output file in the VTK format that can be graphically post processed with VisIt, a 4D visualization/animation open source tool from LLNL.

MODULE NAME

Convert Data To Graph

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

The Convert to Graph tool is used to convert data generated from NCA and Kagan to Timeseries graph data files that can be read by the Graph Viewer module. For a detailed description of the Timeseries graph data format, see section 4. The valid input file formats are

- SBML generated by NCA
- GCM files generated by Kagan
- suggested_interactions.dat files generated by Kagan

MODULE NAME

DBAgent

INSTITUTION

SRI International

WEB SITE

cvs-repository\tools\sri\dbagent\docs\index.html

SUMMARY

The Bio-SPICE DB Agent provides an OAA query service for relational databases. The design of the agent was inspired by the JDBC (Java Database Connectivity) API and provides similar functionality for OAA users. It was developed to support the Bio-SPICE Data Warehouse effort at SRI but it is of generic utility and can be used to access data of any kind.

The current Bio-SPICE DB Agent implementation has been tested using Oracle 8.1.7, PostgreSQL 7.2.1, and MySQL Ver 11.17 Distrib 3.23.49a. A properties file is used to set the JDBC driver and connection URL at runtime so the DBAgent should work with any database for which there is an available JDBC driver. The agent itself is written in Java but like all OAA agents it can service requests from clients written in any language supported by OAA.

Complete source code is provided for the DB Agent as well as extensive API documentation. A complete example client implementation is included that demonstrates the usage of the entire DB Agent API.

MODULE NAME

Exact Stochastic Simulator (ESS)

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/

http://biocomp.ece.utk.edu/tools.html,

download tools and open README.pdf

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UTK/ess.pdf

cvs-repository\tools\utenn\ess

SUMMARY

ESS is a stochastic simulator based on Gillespie's Stochastic Simulation Algorithm. It takes SBML and BioSpreadsheet model files as input and stochastically predicts the time evolution of the chemically reacting species, which it outputs as a Timeseries.

Double clicking the ESS module allows the rate at which data is printed (PrintInterval), the time when the simulation should be complete (EndTime), and random seed (seed) to be configured by the user.

MODULE NAME

Fluxor Computational Analyzer

INSTITUTION

Harvard University

WEB SITE

http://arep.med.harvard.edu/moma/

http://arep.med.harvard.edu/gmc/fba.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Harvard/harvard_tools.pdf

cvs-repository\tools\harvard\Fluxor\Fluxor_usecase_7-04\FluxorPipeline_tar.gz\Fluxor

SUMMARY

The assessment of our ability to extract and understand biological information from genomes lies in the capacity to build computational models and make predictions that can be tested. This use case distribution contains the initial versions of methods for utilizing annotated genomes to construct whole-cell models, as well as analytical tools to make predictions using these models.

There are four major parts to this use case contribution. One is the BioWarehouse from SRI which can be set up to hold reaction datasets that can be analyzed using flux analysis methods. The second part consists of a couple of tools that can produce SBML representations of datasets to be analyzed, one that represents data stored in the BioWarehouse and one that produces an SBML representation of an important flux model that has been published. The third part is a spreadsheet graphical user interface that can be used to prepare models for analysis. The final part consists of two tools that carry out different aspects of flux analysis, Fluxor predicts resulting fluxes under different conditions, and metatool determines elementary flux pathways from available metabolites to final products. In the future we intend to provide a fifth part to this use case that will convert the SBML produced by these analyzers into BioWarehouse databases.

One readily available source of annotated genomes is available from SRI as a collection of databases stored in a Bio-SPICE compatible database system known as the BioWarehouse, which can be downloaded from a link on the Bio-SPICE community home page. Included in this distribution is a perl script called biowarehouse2sbml.pl created by Harvard Medical School and provided as a dashboard analyzer that creates a valid SBML Level 2 file from a BioWarehouse dataset representing the list of metabolic reactions for a particular genome.

In this distribution we provide two sets of dashboard analyzers for converting an SBML file obtained from the BioWarehouse or any other source into a fully specified flux model and making predictions from this flux model using flux balance analysis. One of these is an analyzer that uses a program called Metatool from the Software Biology Workbench provided by the Keck Graduate Institute to figure out elementary flux paths from the metabolites provided to a reaction, typically a biomass production reaction, that describes the overall behavior of the system. We also provide a pair of analyzers developed at Harvard Medical School that are called Fluxor. The first is a graphical interface called FluxorSpreadsheet that enables the user to add some of the features required to turn a reaction list into a complete flux model using a graphical interface based on the BioSpreadsheet developed at the University of Tennessee. It dispatches a

resulting further annotated SBML file to the Fluxor computational analyzer that performs flux predictions. The current version of the user interface already enables a user to set flux limits on selected reactions in the full list and to indicate which fluxes are to be disabled so that the Fluxor computational analyzer can predict fluxes that result from these knockouts. Future versions will enable the user to remove conflicting reactions from a model as well as add additional reactions needed to create a full flux model, including a biomass production reaction to be optimized.

We have also developed at Harvard Medical School a tab-delimited list of all the fluxes and limits, both metabolic reactions and transport fluxes, provided for the JR904 *E. coli* flux model published by Palsson in 2003. With assistance from the Center for the Development of Advanced Computing in Puna, India, Harvard Medical School has written a perl script called writesbmljr904.pl that converts this tab-delimited list into SBML suitable for use by the Fluxo graphical interface and analysis by the Fluxor program. We were not able to verify the completeness and correctness of this representation of the JR904 model using Fluxor prior to submission, but we are providing the tab-delimited file, the writesbmljr904.pl that converts it (following possible necessary changes) into SBML, as well as the SBML file created from the distributed version of the tab-delimited file that can be used as input to the Fluxor graphical interface and program.

The current submission does not include everything indicated in our full use case diagram, nor the full capability of linking everything together that we will ultimately produce. We have not yet developed the sbml2biowarehouse tool that will load the sbml-represented results of running metatool or fluxor into the BioWarehouse, nor have we added the fields in the BioWarehouse structure needed to represent things like flux limits, flux objective functions and flux predictions. The Fluxor Spreadsheet user interface is not yet able to ask metatool to perform elementary flux path analysis and provides only some of the model modification capabilities that are likely to be required to convert most BioWarehouse databases into fully specified flux models. The version of metatool provided in this submission only works on a linux system like the other tools in this use case in very special circumstances we have not yet been able to define. The FluxorPipeline/Metatool/readme.metatool file describes most of the steps necessary to set up Metatool on a linux system, although at this point we have not been able to set it up to work except in a single account on a single machine. It is, however, possible to set up the metatool analyzer and the Software Biology Workbench on a windows system and use it to analyze an SBML file produced by any of the analyzers, and that is also described in the readme file.

MODULE NAME

Fluxor Spreadsheet

INSTITUTION

Harvard University

WEB SITE

http://arep.med.harvard.edu/moma/

http://arep.med.harvard.edu/gmc/fba.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Harvard/harvard_tools.pdf

cvs-repository\tools\harvard\Fluxor\Fluxor_usecase_7-04\FluxorPipeline_tar.gz\FluxorSpreadsheet

SUMMARY

Fluxor Spreadsheet provides a spreadsheet-like interface to allow user to specify nutrient conditions, external metabolites, and gene knockouts, and to display the results of various kinds of flux predictions.

MODULE NAME

GeneCite

INSTITUTION

Walter Reed Army Institute of Research

WEB SITE

http://ftp.bioinformatics.org/pub/wrair/genecite/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/WRAIR/GeneCite.pdf

cvs-repository\tools\wrair\GeneCite

SUMMARY

GeneCite is a generalized query application that allows you to specify sophisticated sets of queries and generates a table of the number of citations found for each query. The table can be presented as a Web page or in a standard spreadsheet format that will allow you to view the full output of only those queries that generate an interesting number of citations.

Currently, GeneCite allows you to submit 'term'-type queries to the U.S. National Library of Medicine's PubMed database. Other kinds of queries and databases may be added in the future.

You provide the application with partial queries in standard ASCII text files. These queries can then be combined in various ways to produce the set of queries that is sent to the database, called a search. The program stores the number of citations returned for each query. The result of a search is a column (one-dimensional) or a table (two-dimensional) of those counts, depending on the type of search selected. The results can be output in a variety of formats. The list of output formats may be enlarged in the future, but currently includes:

- a HTML Web page compatible with standard web browsers,
- a comma-delimited set of hyperlinks suitable for importing into standard spreadsheet applications such as Microsoft Excel,
- a comma-delimited (spreadsheet-compatible) set of numerical counts only, for numerical analysis of the data, which cannot be done on hyperlinks.

The hyperlinks in the first two kinds of file allow you to re-run queries in which you are interested, receiving the full list of citations this time, instead of just the citation counts.

GeneCite has a number of advantages over simply keying queries into a database. In its simplest, one-dimensional, mode, it allows you to type a set of queries once with a simplified syntax and run them repeatedly over a period of time, to look for changes. It also can execute large batches of queries unattended, leaving you free to do something else while it runs. In its two-dimensional modes, it allows you to generate complex combinations of queries automatically. GeneCite also reduces the result of each query to a single value, the number of related citations, while preserving the ability to regenerate any query result simply by clicking on it. Instead of wading through a whole web page for each query, you see a summary of the results of a whole set of queries, so you can choose the ones that are relevant to you.

MODULE NAME

GeneScreen

INSTITUTION

University of California, Los Angeles

WEB SITE

http://www.ee.ucla.edu/~riccardo/GeneScreen

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UCLA/Genescreen

cvs-repository\tools\ucla\genescreen

SUMMARY

GeneScreen is a collection of computational statistic routines, whose objective is to process gene expression data (typically from DNA microarray time-course experiments), extracting significant gene association patterns.

In GeneScreen, the conditional mutual information among genes in a cluster (also known as the co-information) is used as a measure of conditional dependency. The mutual information is used as a scoring metric for its capability of detecting dependencies of high-order (non-linear), as opposed to a simple correlation measure, which is only capable of representing linear dependencies in the data.

For a given microarray assay experiment, all possible unique combinations of three genes are considered and the co-information is used to assign a score to each such combination. GeneScreen includes a set of tools devoted at pre-processing of the transcription data, performing a series of tasks which include pruning the set of genes according to a user defined criterion (e.g. their sample variance), correcting for univariate and bivariate outliers, or accounting for missing values.

MODULE NAME

GeneWays

INSTITUTION

Columbia University

WEB SITE

http://geneways.genomecenter.columbia.edu/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Columbia/geneways.pdf

cvs-repository\tools\columbia\Geneways4BioSPICE_sandbox

SUMMARY

GeneWays is a system for automatically extracting, analyzing, visualizing and integrating molecular pathway data from the research literature. The system focuses on interactions between molecular substances and actions, providing a graphical consensus view on the collected information. GeneWays is designed as an open platform, allowing researchers to query, review and critique the integrated information

The word "GeneWays" is derived from the words "genes" and "pathways." The system was designed with the ambitious goal of automating extraction of information on molecular interactions locked in the text of journal articles. The number of nodes in human molecular networks is measured in hundreds of thousands when all substances (genes, RNAs, proteins, and other molecules) are considered together. These numerous substances can be in turn present or absent in dozens of cell types in humans. Clearly, the complexity is too great to yield to manual analysis. Thus, with the hope of relieving the information overload currently assaulting scientists, we are developing GeneWays, a computer system that integrates a battery of tools for automatic gathering and processing of knowledge on molecular pathways.

GeneWays focuses on molecular interactions pertinent to signal-transduction pathways. In contrast to metabolic pathways, which mostly deal with tremendously diverse chemical alterations of relatively small molecules, signal-transduction pathways are relatively poor in chemical mechanisms and predominantly involve "switch-on" and "switch-off" interactions among large molecules, such as genes and proteins. In an article describing a

signal-transduction pathway statements that "protein A binds protein B," "protein C phosphorylates protein D," and "protein E activates gene F" are seen frequently.

GeneWays extracts relations between substances or processes from free text journal articles. At present, over 250,000 articles have been analyzed by the system. GeneWays processes journal articles as follows:

The Term Identifier module identifies biologically important concepts in the text, such as names of genes, proteins, processes, small molecules, and diseases.

The GENIES parser derives a semantic representation of the relations present in a sentence (if any). The output of GENIES is a semantic parse tree.

The Simplifier module compiles the semantic trees into simple binary statements. An example of a simple binary statement is "interleukin-2 binds interleukin-2 receptor". This statement links two substances, interleukin-2 and interleukin-2 receptor, with action "bind".

These statements are stored in the GeneWays Database.

MODULE NAME

Get Column from Timeseries

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

This module extracts a column of data from a Timeseries. The module takes a Timeseries as input and a produces a Timeseries as output. Double-clicking the module allows configuration of the parameter, “Column Name”, which specifies the name of the column to extract.

MODULE NAME

Get Row from Timeseries

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

This module extracts rows of data from a Timeseries. The module takes a Timeseries as input and a produces a Timeseries as output. Double-clicking the module allows configuration of the parameters, “StartRowIndex” and “EndRowIndex” which specifies the range of rows to extract. The module assumes that the first row of data is the header, therefore the data immediately following the header row is called row 1.

MODULE NAME

Graph Viewer

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

The Graph Viewer is used to display interactions between multiple genes or proteins. It takes a Timeseries graph as input (see section 4 for details on this format). Selecting different items on the left panel will cause different nodes of the graph to be displayed. Multiple nodes can be selected for display by holding the SHIFT or CTRL keys while selecting items. The graph viewer will attempt to automatically place nodes to display a nice graph, but nodes can be rearranged by clicking and dragging them. The start/stop button toggles the automatic placement of nodes. Clicking a node causes it and its neighbors to be highlighted.

MODULE NAME

Graphviz

INSTITUTION

Lawrence Berkeley National Laboratory

WEB SITE

http://genomics.lbl.gov/index.html

http://biospice.lbl.gov/home.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/LBL/pathwaybuilder.pdf

SUMMARY

The graphviz support module provides capabilities to generate transform the biochemical networks within an SBML file into a Graphviz DOT (graph) file. This graph file can then be manipulated by Graphviz to allow visualization of the network. This is useful for laying out SBML models that may not have layout information currently associated with them. It is necessary for the graphvizpathway module, which automatically lays out SBML files and imports them into PathwayBuilder.

MODULE NAME

Homolog Finder

INSTITUTION

Lawrence Berkeley National Laboratory

WEB SITE

http://genomics.lbl.gov/index.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/LBL/homolog_finder.pdf

cvs-repository\tools\arkin\homologuefinder\index.html

SUMMARY

The homolog finder accepts a PathwayBuilder pathway in which entities in the pathway where placed there through a query of BioDB for proteins/DNA from a given organism. Homolog finder takes such a pathway and allows all molecules in the pathway with a BioDB identifier to be searched against another organism. Orthologous proteins in the second organism are derived using BLAST. An ortholog is defined informally either as a bi-directional best hit from blast of organism 1 to organism 2 and vice versa or as uni-directional best hit. The output of Homolog finder is a new PathwayBuilder pathway in which, for each entity with an ortholog, the database id of the entity is changed to point to the new organism's version of the gene and the color of the graphic represents how close an ortholog it is.

MODULE NAME

Hybrid Automata Symbolic Reachability Tool

INSTITUTION

Stanford University

WEB SITE

http://cherokee.stanford.edu/~ronojoy/biospice/

cvs-repository\tools\stanford\SymbolicAnalysisToolbox\SymbolicAnalysisToolbox_9-04\hybridtool_zip\doc1.doc

SUMMARY

Hybrid performs parameter identification of biological systems through using piecewise affine hybrid automata.

Hybrid Automata Symbolic Reachability Tool computes sets of initial protein concentrations from which a biologically interesting steady state can be achieved and indicates automatically whether additional constraints are necessary for a biologically interesting steady state event to happen.

MODULE NAME

IcDNA

INSTITUTION

University of California, Los Angeles

WEB SITE

http://receptor.seas.ucla.edu/lcDNA/

http://receptor.seas.ucla.edu/lcDNA/instructions.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UCLA/lcDNA.pdf

cvs-repository\tools\ucla\lcdna.ver.32\lcDNA

SUMMARY

lcDNA is a software program for calculating statistical confidence levels of DNA microarray data based on a Markov Chain Monte Carlo simulation.

MODULE NAME

JDesigner

INSTITUTION

Keck Graduate Institute

WEB SITE

http://sbw.kgi.edu/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/KGI/kgi_docs.pdf

cvs-repository\tools\kgi

SUMMARY

As a contribution to the Bio-SPICE project, we have developed an optimization module, for the task of fitting kinetic rate constants to time series concentration data. The algorithms use, both local searches, such as the Levenberg-Marquardt, and simplex, as well as global search methods such as simulated annealing and real coded genetic algorithms.

In this document we provide a basic description of the software components that were used to build the optimization module. The details of the methods used, parameter estimation issues and a general description of the optimization modules, can be found in (Chickarmane S. V et. al. 2004)

Software Components

In the following we will briefly describe the software components.

- SBW: SBW is a software infrastructure that permits applications written in different languages to communicate with each other (http://www.sbw-sbml.org/.)
- *Designer: JDesigner is a model editing tool, it can be used to display the model (Sauro et. al 2003).
- Jarnac: Jarnac is a simulation engine (Sauro, 2000).
- Matlab dll's: These dll's contain the optimization algorithms.
- OptController: The Optimization Controller: this is the optimization GUI controller, which provides the user interface.
- DataGen: The Data Generator, which is used in conjunction with the OptController, to simulate noisy data.
- Chaperone: The bridge between SBW and the compiled Matlab dll' s.

MODULE NAME

JigCell Tools

INSTITUTION

Virginia Polytechnic Institute and State University

WEB SITE

http://jigcell.biol.vt.edu/index.html

http://jigcell.biol.vt.edu/jigcell/docs/JigCell/Building.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/VaTech/jigcell.pdf

cvs-repository\tools\vatech\jigcell

SUMMARY

JigCell is a joint effort by members of the Departments of Biology and Computer Science at Virginia Tech. Our goal is to advance the state of the art for modeling in Systems Biology.

A major challenge of contemporary cell biology is to understand cell physiology from its underlying molecular regulatory networks. These networks are complex, containing many components interacting with one another through various positive and negative feedback loops. The dynamical consequences of these feedback loops are dauntingly complex, and it is not possible to understand them by intuitive arguments alone. Mathematical models are needed to describe the network precisely, to analyze their interactions rigorously and to provide accurate simulations that can be compared with experimental observations in quantitative details.

Because the cell cycle underlies the growth, development, and reproduction of all living organisms, knowledge about its control is central to cell biology and has potential applications in the health care and pharmaceutical industries. We want to develop novel, computational tools, with user-friendly interfaces, for studying complex biochemical regulatory systems in general, and the cell cycle control system in particular.

Our primary results are of two types. We create and distribute models of the cell cycle. And we create and distribute software that supports modelers of biochemical reaction pathways. Our tools currently include a model builder, run manager, comparator, and automatic parameter estimator. We hope to soon add a numerical bifurcation analysis package.

Our team provides a unique synergy between biologists and computer scientists. Our philosophy is that this synergy is an important driver for advancing the state-of-the-art in systems biology. By creating advanced software tools, computer scientists can help biologists to become more productive. By having the computer scientists work directly with practicing biology modelers, the biologists can guide the computer scientists to write software that is worthwhile.

JigCell Tools includes several capabilities in one package. The Model Builder builds and edits SBML models using a spreadsheet interface, reducing models by finding conservation relations in the equations. The Run Manager defines an ensemble of runs that specify how to simulate SBML models with various changes to parameter and initial condition values. XPP provides access to Bard Ermentrout's XPP simulator. Biopack provides access to the numerical integration routines of LSODAR. The Comparator executes an ensemble of runs, applies data

transformations to the simulation output, and determines how well the model matches the collected experimental results it is attempting to reproduce. The Parameter Estimator fits model parameters by applying local and global optimizers to find values that bring the model and experimental results into close agreement. Compare2 allows the user to visualize the change in the fit of a model to experimental results as a model is developed over time. The Project Manager stores and organizes data files used by other tools into logically related projects.

MODULE NAME

JournalMine ActionQuery

INSTITUTION

Columbia University

WEB SITE

http://textmine.cu-genome.org/html/index.html

SUMMARY

The JournalMine Action Query module is a wrapper around a webservice which queries the Columbia Genome Center's search engine.

MODULE NAME

Karyote Cell Analyzer (KCA)

INSTITUTION

Indiana University

WEB SITE

http://biodynamics.indiana.edu/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Indiana/KCA.pdf

cvs-repository\tools\uindiana\KaryoteAnalyzer

SUMMARY

The Karyote Cell Analyzer (KCA) is based on the principles of chemistry and physics as formulated for single and multiple cell systems. It is designed for fundamental studies into the workings of a cell and for applications of this understanding in drug and vaccine design, treatment optimization, refined diagnoses of complex diseases like cancer and biotechnical process engineering. The input to KCA is the network of biochemical processes and rate/equilibrium data, intracellular architecture, membrane transport properties, initial cell state and conditions in the extracellular system. The output of KCA is the timecourse of all chemical species in all compartments (and all cells for a multicellular simulation). More can be found about KCA at biodynamics.indiana.edu.

KCA is the first module from the Center for Cell and Virus Theory (CCVT)being made available through the Bio-SPICE platform so that it can be used as part of workflows involving other Bio-SPICE modules. For assistance with KCA please contact us at ccvt@indiana.edu or review our website, biodynamics.indiana.edu. The dual availability of this and other modules through Bio-SPICE and our website ensures both compatibility with other software in the Bio-SPICE system and our most up-to-date science that Bio-SPICE has not yet been designed to accommodate (e.g. multi-dimensional cell modeling and molecular scale phenomena).

MODULE NAME

Karyote Genome Analyzer (KAGAN)

INSTITUTION

Indiana University

WEB SITE

http://biodynamics.indiana.edu/

http://biodynamics.indiana.edu/CellModeling/AboutKaryote.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Indiana/KAGAN.pdf

cvs-repository\tools\uindiana\Kagan

SUMMARY

KAGAN (KAryote Genome ANalyzer) is a software package that receives raw time series microarray data, the list of factors that regulate each gene, and yields the timecourse of thermodynamic activities within the nucleus or prokaryotic cell. Software packages that could be used to provide input to KAGAN include PAINT (developed at Thomas Jefferson University), which gives gene/transcription factor relationships. The results of KAGAN are provided graphically in terms of the predicted timecourse of transcription factor activities. The latter provides information that can be used to detect errors in the gene control network. For example, if a gene is upregulated by one transcription factor and unaffected by all others then a change in transcription factor activity should be well correlated with a microarray response for that gene; if this is not the case, the network requires correction.

KAGAN is built on an entropy maximization principle. Entropy is a quantitative measure of uncertainty. It is used in our methodology to provide a framework for data/cell model integration. The information theory approach of Sayyed-Ahmad et al. (2003) and a transcription kinetic model similar to that in Weitzke and Ortoleva (2003) are used.

KAGAN is being provided as a remote computational service. We do not retain the user's microarray data. As the user does not provide gene or transcription factor names, the nature of the time series experiment, and research objectives remain proprietary.
In this Use Case we demonstrate the integration of computations on the Dashboard and at a remote site. The objective is to input time series microarray data and the structure of the gene control network and have statistical analysis and model computations run automatically. The raw input microarray data is channel normalized, quality filtered and multiple spots/multiple slides averaged.

The resulting pre-processed data is automatically used by our Karyote Genome Analyzer(KAGAN) and the time series of intra-nuclear (or intra-bacterial) transcription factor (TF) timecourses are computed. One may then plot the latter on the Dashboard.

To derive the maximum information from microarray and other bioanalytical data, we believe that three elements must be integrated into one system:

- a kinetic cell biomic model with its richness of physico-chemical laws and cell biochemical processes, but admitting its incompleteness and inaccuracies;

- methods for utilizing a set of microarray, spectral and other data with their rich multiplex character but admitted inaccuracies, incompleteness and difficulty in interpretation; and
- a procedure for automatically integrating (1) and (2) so that the strengths of one compensates for the gaps/uncertainties in the other.

Cell kinetic models, key to our approach, have been developed over the past 50 years. Our approach is kinetic, avoiding restrictions to its applicability that are imposed by steady-state models. Thus our approach is viable for cell cycle, oscillatory metabolic phenomena, response to changes in the extra-cellular medium, developmental phenomena, nuclear transplantation and transitions to cancer and other abnormalities. However, kinetic cell models are still in an incomplete state. Our approach allows an incomplete model to be integrated with data in a manner whereby the data complements the model and, despite its incompleteness, the model facilitates the interpretation of the data, all in a seamless, automated fashion. KAGAN illustrates our approach for time series microarray data. The type of cell model we use determines the richness of information that can be obtained from multiplex bioanalytical data. In KAGAN we use a transcription kinetic model integrated with time series data to perform cell simulations despite model incompleteness. To this end we utilize a transcription model and microarray time series data to construct the most probable timecourse of intra-nuclear TF thermodynamic activity. With this information, the manner in which TFs coordinate multiple gene responses to applied stress, inserted genes or other influence can be understood. We emphasize that this TF dynamics is obtained even though the translational, post-translational and metabolic processes are not accounted for in the model↓i.e. through our information theory approach one may run a model even though for some key variables (here dynamically varying TF activities) no governing equations are provided.

The benefit of our remote KAGAN service is that we are continuously improving our algorithms and the level of the detail of the information we extract. This is essential in a fast evolving field wherein it is difficult for SBML and other protocols to adapt to new science.

MODULE NAME

Merge Timeseries Columns

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

Use this tool to merge two Timeseries data set together. This merges the two data sets together horizontally, appending the columns of one data set to another. The two data sets must have the same number of rows

MODULE NAME

Merge Timeseries Rows

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

Use this tool to merge two Timeseries data set together. This merges the two data sets together vertically, appending the rows of one data set to another. The two data sets must have the same number of columns.

MODULE NAME

MetaCluster

INSTITUTION

Thomas Jefferson University

WEB SITE

http://www.dbi.tju.edu

http://www.dbi.tju.edu/tools/biospice/usecase/usecase.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/TJU/tju_tools.pdf

cvs-repository\tools\tju\metacluster

SUMMARY

The biological problem we study is unraveling the transcriptional regulatory networks underlying various cellular functions, specifically in mammalian systems. System-wide high throughput gene/protein expression data enables us to analyze gene regulation, in particular co-regulation, at a system level. We have developed various tools that enable biologists to integrate functional genomics data, for example from microarray-based gene expression analysis, with genomic sequence data to carry out transcriptional regulatory network analysis (TRNA). For more information, please refer to this presentation.

A large diversity of clustering algorithms are available for data analysis, all of which generate results which are specific to the algorithm used or even to the individual iteration. MetaCluster was developed to help biologists mine several different clustering results for those data relationship insensitive to the clustering methods used. Metaclustering provides a computational tool which co analyses diverse clustering results to highlight the relationships that are stable across algorithms. These method-independent co clustering results provide the strongest evidence for biological significance.

For more information on Metacluster algorithm, please refer to this publication (PDF). The MetaCluster Toolbox included in this Dashboard contribution features a GUI that takes Timeseries as input (such as microarray expression data), allows the user to cluster the data interactively, and produces a SBML2 output of the clustering results.

We have developed a few Analyzers for the Bio-SPICE Dashboard, most notably PAINT. Older Dashboard Analyzers from TJU - MetaCluster and CloneUpdater - have been phased out with a hope to be replaced by better tools. Some of the functionality is incorporated into PAINT to enable the Bio-SPICE community use PAINT in a meaningful way.

MODULE NAME

MetaTool

INSTITUTION

Keck Graduate Institute

WEB SITE

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/KGI/kgi_docs.pdf

http://public.kgi.edu/~vchickar/Optimization/optimizationfinal.pdf

cvs-repository\tools\kgi\kgi\Metatool

SUMMARY

As a contribution to the Bio-SPICE project, we have developed an optimization module, for the task of fitting kinetic rate constants to time series concentration data. The algorithms use, both local searches, such as the Levenberg-Marquardt, and simplex, as well as global search methods such as simulated annealing and real coded genetic algorithms. In this document we provide a basic description of the software components that were used to build the optimization module. The details of the methods used, parameter estimation issues and a general description of the optimization modules, can be found in (Chickarmane S. V et. al. 2004).

MODULE NAME

MIAMESpice

INSTITUTION

University of California, Los Angeles

WEB SITE

cvs-repository\contrib\ucla\Miame-SPICE\MiameSPICE_usecase_7.04\me_v1_0_1_window_zip\README.txt

SUMMARY

MIAME Spice packages raw and normalized datafiles from a set of related microarray experiments, saving all associated data from an experiment (or set of experiments) into one archive file. Users can also enter experimental annotations, array design information, and array design files.

MODULE NAME

Monod

INSTITUTION

The Molecular Sciences Institute

WEB SITE

http://monod.molsci.org/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/MSI/monod.pdf

cvs-repository\tools\molsci

SUMMARY

Monod Desktop is a desktop application written in Java as a companion software to web Monod. The main objectives are to provide

1. Easy navigation
2. Easy data submission
3. Easy data mining

Monod Desktop is currently at the prototype phase. The prototype has primarily focused on the following Monod Data Entities :

1. Species
2. Process
3. Model
4. Note annotation
5. Literature annotation

The prototype has included some basic functionality as a mean to gather feedback while development for the next release is underway.

Monod Desktop currently has five main components:

Text-based Data Mining: Species process model, annotation can be searched.

Note Annotation: Annotation (e.g., comments or file attachments) can be created, edited and queried.

Literature Annotation: Literature (journal articles and file attachments) can be created and queried.

User Workspace: Users can create folders within the assigned workspace. Favorite items or work in progress items can be put into the workspace or a workspace folder for quick access.

Model Editor: Models can be created, saved, edited and viewed graphically.

MODULE NAME

Network Component Analysis (NCA)

INSTITUTION

University of California, Los Angeles

WEB SITE

http://www.seas.ucla.edu/~liaoj/NCA1.htm

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/UCLA/nca/INSTALL.txt

cvs-repository\tools\ucla\NCA_analyzer

SUMMARY

NCA (Network Component Analysis) is a Matlab package at whose core is a routine for the analysis and parameter estimation of gene transcriptional networks. NCA requires two separate types of inputs. The first is a description of the available connectivity information between genes and transcriptional factors (for example the result of binding site analysis), which is given as a boolean adjacency matrix. The second is a set of gene expression level time-courses, of the type obtainable for example by DNA microarray experiments.

MODULE NAME

NYUMAD

INSTITUTION

New York University

WEB SITE

http://bioinformatics.nyu.edu/Projects/nyumad/

http://bioinformatics.nyu.edu/Projects/index.shtml

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/NYU/NYUMAD.pdf
cvs-repository\tools\nyu\nyumad\nyumad-rel-2.0

SUMMARY

Many problems in functional genomics are being tackled using Microarray Technology. While this approach holds much promise for answering open questions in Biology, they pose significant problems from the Data Management, Analysis, and Distributed Collaboration point of view. The NYU Bioinformatics Group has been involved in several projects that use Microarray technology. Among these, Nitrogen Pathway analysis in Arabidobpsis (with the NYU Biology Dept.), and Cancer related cell signaling using different cell lines (with CSHL).

To address the needs of these collaborative research groups and others, we have developed the NYU Microarray Database (NYUMAD). NYUMAD functionality ranges from the storage of the data in relational data base management systems to front-end capabilities for the presentation and maintenance of the data. Collaborating groups can share data and analysis results immediately and selectively, with well defined “publishing privileges”.

The database is a unified platform to understand the microarray gene expression data. The data can be output to a wide class of clustering algorithm, based on various "similarity measures" and grouping approaches. Particularly, we have developed a new statistically-robust similarity measure based on James-Stein Shrinkage estimators and provided a Bayesian explanation for its superior performance. We will also incorporate statistical tests for validation and measuring the significance (e.g., jackknife and bootstrap tests).

In the development of NYUMAD we adhered to emerging and well known standards and provide avenues for extensibility. Most of the underlying DB schema design follows closely the specifications put forth by the Microarray Gene Expression Database group(MGED.) The functionality of the NYUMAD system is summarized hereafter.

Microarray data is stored in relational data base management systems (RDBMS) using a database schema based on the MAML (Microarray Mark-up Language) specification.

Data is served to "clients" over the Internet. Clients can be the NYUMAD Java GUI applet we provide, or custom-built user programs, which read and write simple CSV files or MAML and MAGE-ML files retrieved using a simple HTTP protocol requests. In the case of the NYUMAD Java applet, data retrieval is transparent to the user. Data submissions for updating existing data or inserting new data can also be made using the NYUMAD Java applet client, or by custom-built user programs, or HTML forms that access directly the server. As with data retrieval, the

GUI front-end capabilities of the NYUMAD Java applet make data submission transparent to the user.

We took extreme care in making the system "distributable" from the start, by clearly defining a three tiered architecture that allows us to concentrate on different aspects of the design.

MODULE NAME

NYUSIM

INSTITUTION

New York University

WEB SITE

http://bioinformatics.nyu.edu/Projects/index.shtml

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/NYU/NYUSIM.pdf

cvs-repository\tools\nyu\nyumad\nyumad-sim_9-04

SUMMARY

At the intersection of Biology and Computer Science there is a fertile field where simulations of biochemical systems are used to produce hints for experimenters and explanations for biological hypotheses. Many of these simulations have a very simple structure based on a collection of vectors taken at a series of time points. The difference between these traces is usually in the meta-information attached to them in the form of parameter sets and explanations of the methods used to obtain them.

Manipulating these traces and allowing to catalog them in a consistent way is one of the problems facing collaborative research in Biology.

To address the needs of these collaborative research groups and others, we have developed the NYU Simulation Database (NYUSIM,) derived directly from NYUMAD. NYUSIM functionality ranges from the storage of the data in relational data base management systems to front-end capabilities for the presentation and maintenance of the data. Collaborating groups can share data and analysis results immediately and selectively, with well defined “publishing privileges”.

Simulation data is stored in relational data base management systems (RDBMS) and communication to and from the database is achieved via a number of interfaces (e.g. Matlab,) and well specified formats, like the DARPA Bio-SPICE Time Series format.

Data is served to "clients" over the Internet. Clients can be the NYUSIM Java GUI applet we provide, or custom-built user programs, such our Java API (used by Matlab) which retrieves Bio-SPICE Time Series formatted files using a simple HTTP protocol requests. In the case of the NYUSIM Java applet, data retrieval is transparent to the user.

MODULE NAME

Octave Bridge

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/UTK/octave_bridge.pdf

cvs-repository\tools\utenn\OctaveBridge

SUMMARY

This module serves as a bridge to the powerful data post-processing tool Octave. To use it, you must first install Octave, which is available for both Windows and Linux at www.octave.org.

The module takes Timeseries data as input along with the following parameters, which can be configured by double-clicking the module.

Command – The command or set of commands (separated by semicolons) that you wish to run.

OutputName – The name of the output variable you wish to pass on as output.

OctavePath – The path to the octave executable (this is greatly simplified if you add octave to your PATH).

The output is a Timeseries data set that represents the results of the command.

The input Timeseries must consist of a header row followed by numeric data. The module starts the octave program and creates variables to represent each column. The first element of each column (the header) is used as the variable name. The command specified by the "Command" parameter is then executed. The specified "OutputName" is used to determine what data to pass back to the user.

MODULE NAME

Oscill8 Tools

INSTITUTION

Virginia Polytechnic Institute and State University

WEB SITE

http://oscill8.sourceforge.net/

SUMMARY

Oscill8 is a suite of tools for analyzing dynamical systems which concentrates on understanding how the dynamical behavior depends on the parameters using bifurcation theory and reaction network theory.

MODULE NAME

PAINT

INSTITUTION

Thomas Jefferson University

WEB SITE

http://www.dbi.tju.edu/dbi/tools/paint/

cvs-repository\tools\tju\PAINT

SUMMARY

Promoter Analysis and Interaction Network Toolset (V 3.5)

Highly parallel gene expression analysis has led to analysis of gene regulation, in particular co-regulation, at a system level. PAINT was developed to provide the biologist a computational tool to integrate functional genomics data, for example from microarray-based gene expression analysis, with genomic sequence data to carry out transcriptional regulatory network analysis, TRNA. TRNA combines bioinformatics, used to identify and analyze gene regulatory regions, and statistical significance testing, used to rank the likelihood of the involvement of individual transcription factors, with visualization tools to identify transcription factors likely to play a role in the biology under study. In addition this tool can output results in several different formats for use with modeling and simulation tools.

Given a list of genes, PAINT can:

Fetch potential promoter sequences for the genes in the list.

Find Transcription Factor (TF) binding sites on the sequences.

Analyze the TF-binding site occurrences for over/under representation compared to a reference.

Generate multiple visualizations for these analyses

MODULE NAME

Pathway Builder

INSTITUTION

Lawrence Berkeley National Laboratory

WEB SITE

http://genomics.lbl.gov/index.html

http://biospice.lbl.gov/PathwayBuilder/

http://biospice.lbl.gov/home.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/LBL/pathwaybuilder.pdf

cvs-repository\tools\arkin\pathwaybuilder

SUMMARY

This release of Berkeley Bio-SPICE provides software tools that can be used to graphically represent a biological pathway, formally define conceptual models of processes in the pathway, define mathematical representations of those models, and produce representations of those models that can be used to execute simulations using Matlab or SBML compliant simulators. The PathwayBuilder tool (see "doc/manual.html") provides the user with the capability to draw and annotate a pathway. In addition, the current version of the PathwayBuilder includes interfaces to a database of biological data (BioDB) containing process definitions and model associations (ProcessDB). The file INSTALL.txt describes how to instantiate these databases using the schemata provided (see the "db" directory). The ProcessBrowser is a stand-alone tool that creates new processes at different levels of granularity, from general to detailed, and stores them in ProcessDB. The PathwayBuilder uses the process information in ProcessDB to instantiate processes in pathways. Because of the integration of the ProcessDB into the PathwayBuilder, a working and populated version of the ProcessDB must be available to the PathwayBuilder for the PathwayBuilder to run. Information on how to create the ProcessDB is provided in INSTALL.txt. The ModelDBTool provides the user with the ability to associate mathematical models implemented as Java classes with processes in the ProcessDB. New models can be created by extending the abstract base class provided with the source code distribution ("src" directory).

MODULE NAME

PlotML Translator

INSTITUTION

SRI International

MODULE NAME

PtPlot

INSTITUTION

SRI International

MODULE NAME

QIS D2

INSTITUTION

Quantum Intelligence Inc.

WEB SITE

http://www.quantumii.com/qi/qisd2.html

cvs-repository\tools\qis

SUMMARY

QIS D2 is designed to predict drug characteristics such as toxicity and efficacy. QIS D2 is a product created from a DARPA funded SBIR project. A QIS D2 data model is successfully trained, tested, and validated on experimental data sets for predicting the potential in vivo effects of drug molecules in biological systems. Of particular interest are effects arising from chemical and biological agents and pathogens. QIS D2 is interoperable with DARPA Bio-SPICE.

The workflow and visualization functions are designed to be easy to use by biochemical professionals. This product has been released with DARPA's Bio-SPICE and also with IBM partners' World. Two commercial pharmaceutical firms are currently evaluating it. We are also collaborating with SRI, WRAIR, USAMRIID, AFIT, and more than 20 other universities and labs nationally.

MODULE NAME

Reactcome

INSTITUTIONS

Cold Spring Harbor Laboratory

The European Bioinformatics Institute

The Gene Ontology Consortium

WEB SITE

http://www.reactome.org/

SUMMARY

The Reactome project is a collaboration among Cold Spring Harbor Laboratory, The European Bioinformatics Institute, and The Gene Ontology Consortium to develop a curated resource of core pathways and reactions in human biology. The information in this database is authored by biological researchers with expertise in their fields, maintained by the Reactome editorial staff, and cross-referenced with the sequence databases at NCBI, Ensembl and UniProt, the UCSC Genome Browser, HapMap, KEGG (Gene and Compound), ChEBI, PubMed and GO. In addition to curated human events, inferred orthologous events in 22 non-human species including mouse, rat, chicken, zebra fish, worm, fly, yeast, two plants and *E. coli* are also available.

We are building a knowledgebase of biological processes in H. sapiens, called Reactome, located at http://www.reactome.org. Reactome is a process-level annotation of key topics in human biology. Unlike an online journal, Reactome is designed to be a bridge between the conventional literature and the growing electronic holdings in genomic databases such as UniProt, NCBI databases, Genbank and Ensembl. In that sense, Reactome is a dual purpose project: on one hand, it is meant for the specialist and the generalist biologist alike who can peruse it like a textbook, while on the other hand it will serve as a treasure trove for the bioinformaticist who can use reasoning logic for analysis of biochemical pathways. Reactome is unique in that it focuses on complete pathways in a top down fashion. All the software related to Reactome is open source, and the data too can be freely downloaded. You can read our recent Press Release at: http://www.cshl.org/public/releases/reactome.html

MODULE NAME

Sample Timeseries

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

This analyzer samples a Timeseries data set at a particular rate. The input is a Timeseries and a parameter specifying the sampling rate. The output is the sampled Timeseries data set.

MODULE NAME

SBW-Matlab

INSTITUTION

Keck Graduate Institute

WEB SITE

http://sbw.kgi.edu/index.htm

http://public.kgi.edu/~cwellock/software.htm

cvs-repository\tools\kgi\sbwmatlab\sbwmatlab_9-04\sbwmatch-09062004_zip\SBWMATCH\docs\SBW_MATLAB Interface.doc

SUMMARY

One of the most popular tools for scientific computing today is MATLAB. While C, FORTRAN, and Perl have their advantages, few tools can turn high-level mathematical thought into high-speed computation as well as MATLAB. MATLAB has a few weaknesses however: chief among these is the difficulty in integrating MATLAB programs with other software. This is particularly chafing in a field like systems biology, where the ability to write interchangeable "software components"—simulators, optimizers, and the like—is sorely needed.

The tools presented in this guide help to solve this problem. SBW is a robust and well-established framework for building systems biology software components; now MATLAB users can use and create these components themselves. Existing SBW modules can be accessed, and new ones can be created entirely inside of MATLAB. Better yet, modules created in MATLAB can be distributed freely, even to users who do not have MATLAB themselves.

MODULE NAME

Strongly Connected Component (SCC) Finder

INSTITUTION

Lawrence Berkeley National Laboratory

WEB SITE

http://genomics.lbl.gov/index.html

http://biospice.lbl.gov/home.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/LBL/scc.pdf

SUMMARY

The Bio-SPICE Strongly Connected Component (SCC) finder

The motif finder is a simple pathway graph motif searcher. There is no theoretical reason why graphical motifs in pathways mean very much though there has been a great deal of literature that tries either to assign ranges of function to such motifs or assigns evolutionary meaning to them. In any case, it is clearly important to be able to enumerate all the feedback loops in a network. The motif finder program is fairly general. However, the current interface only allows searching for SCC's (Strong connected components). An SCC in a graph is a disjoint subset of components of that graph for which any two of those components are connected by a path from the first to the second and back again. This is an interesting substructure because components that are so strongly interlinked and involved in such feedback are very likely to belong to a common function and control structure.

MODULE NAME

Sensitivity Analyzer

INSTITUTION

Lawrence Berkeley National Laboratory

WEB SITE

http://genomics.lbl.gov/index.html

http://biospice.lbl.gov/home.html

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/LBL/sensitivity_analyzer.pdf

SUMMARY

The sensitivity analyzer does a response surface analysis of a model. In this analysis, uncertainty in the model parameters is represented by specifying a minimum, nominal, and maximum value of each parameter. The sensitivity tool then allows the user to choose a feature of the model behavior, such as the time to reach a given concentration, the maximum value of a concentration, or a number of other features and the tool chooses an efficient experimental design for simulations under different parameter assumptions to derive an estimate of the dependence of the feature value on each parameter. The result is returned as a graph of the first-order dependencies.

MODULE NAME

Simpathica

INSTITUTION

New York University

WEB SITE

http://bioinformatics.nyu.edu/Projects/Simpathica/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/NYU/simpathica.pdf

cvs-repository\tools\nyu\simpathica

SUMMARY

Simpathica is a set of tools capable of simulating and analyzing biological pathways. It consists of a front end and a back end. The front end is a pathway editor which generates a set of Ordinary Differential Equations, which are in turn simulated using Octave. The back end, also known as XSSYS, is a temporal logic analysis tool which can answer queries about the time course behavior of a set of pathways. A recent addendum to the XSSYS back-end is a Natural Language Interrogation interface that can be used in lieu of the pure Temporal Logic Query system.

The infrastructure of Simpathica is based on the OAA architecture from SRI and on the NYUSIM database.

Simpathica has been built within the scope of our project Algorithmic Tools and Computational Frameworks for Cell Informatics.

MODULE NAME

SOS Tools

INSTITUTION

California Institute of Technology

WEB SITE

http://www.cds.caltech.edu/sostools/

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/CalTech/sostools.pdf

cvs-repository\tools\caltech\SOS_Tools

SUMMARY

SOSTOOLS can be used to specify and solve sum of squares polynomial problems using a very simple, flexible, and intuitive high-level notation. Currently, the SOS programs are solved using SeDuMi or SDPT3, both well-known semidefinite programming solver, with SOSTOOLS handling internally all the necessary reformulations and data conversion.

SOSTOOLS is a free, third-party MATLAB1 toolbox for solving sum of squares programs. The techniques behind it are based on the sum of squares decomposition for multivariate polynomials [4], which can be efficiently computed using semidefinite programming [24]. SOSTOOLS is developed as a consequence of the recent interest in sum of squares polynomials [12, 13, 20, 4, 18, 10, 9], partly due to the fact that these techniques provide convex relaxations for many hard problems such as global, constrained, and boolean optimization.

Besides the optimization problems mentioned above, sum of squares polynomials (and hence SOSTOOLS) find applications in many other areas. This includes control theory problems, such as: search for Lyapunov functions to prove stability of a dynamical system, computation of tight upper bounds for the structured singular value μ [12], and stabilization of nonlinear systems [16]. Some examples related to these problems, as well as several other optimization-related examples, are provided and solved in the demo files that are distributed with SOSTOOLS.

MODULE NAME

Symbolic Analysis Toolbox

INSTITUTION

Stanford University

WEB SITE

https://biospice.org/modules/Static_Docs/data/ModuleDocumentation/Stanford/hybrid_doc.pdf

cvs-repository\tools\stanford\SymbolicAnalysisToolbox

SUMMARY

The physical properties of the biological processes involved in cell signaling make it well-suited for hybrid systems modeling, using a two-level hierarchical framework. At a low level, the spatio-temporal evolution of protein concentration dynamics are governed by linear differential equations; the second tier consists of the logic network that activates or deactivates the continuous dynamics through discrete switches. The variables of the logic network are the protein concentration metrics from the lower level. Thus, the two levels are strongly interconnected, yet the system as a whole is more amenable to analysis than is a general nonlinear system.

The tools implement an abstraction algorithm that has been developed to compute backward reachable sets of states, which are completely expressed in terms of the system's symbolic parameters. These backward reachable sets, when computed for the steady states of the system, represent regions of state space that are guaranteed to converge to one particular steady state or the other.

Biologically, this implies that one can identify sets of initial protein concentrations from which a biologically interesting steady state can be achieved. For example, a question that can be answered by constructing the reachable set is: "if concentration of protein A is greater than protein B in neighboring cells, which steady state will the cell system converge to?" The importance of the method lies in the fact that specific questions can be asked about the behavior of the system, and the results will indicate automatically whether additional constraints are necessary for a biologically interesting steady state event to .

MODULE NAME

Timeseries to Text

INSTITUTION

University of Tennessee, Knoxville

WEB SITE

http://biocomp.ece.utk.edu/tools.html

Download: utkornltools/README.pdf

SUMMARY

This analyzer converts an input Timeseries data set to tab-delimited text. You can then import this data into another tool, like Microsoft Excel.

MODULE NAME

Vibrio

INSTITUTION

University of Pennsylvania

WEB SITE

https://biospice.org/index.php?module=Static_Docs&type=user&func=view&f=ModuleDocumentation/UPenn/vibrio_readme.txt

cvs-repository\tools\upenn\vibrio

SUMMARY

The bacteria of the vibrio fisheri employ negative feedback when regulating bio luminescence. This program provides a way to model the transcription of the proteins involved in the process as well as the complexes that control the transcription. The key variables are mRNA, LuxR, Lux I, Lux A/B/C/D/E. autoinducer, and the complex Co.